Importance Sampling Techniques in Neural Detector Training

José L. Sanz-González and Diego Andina

Universidad Politécnica de Madrid (Dpto. SSR), ETSI de Telecomunicación-UPM,
Ciudad Universitaria, 28040 Madrid, Spain
{jlsanz, andina}@gc.ssr.upm.es

Abstract. Importance Sampling is a modified Monte Carlo technique applied to the estimation of rare event probabilities (very low probabilities). In this paper, we propose and develop the use of Importance Sampling (IS) techniques in neural network training, for applications to detection in communication systems. Some key topics are introduced, such as modifications of the error probability objective function, optimal and suboptimal IS probability density functions (biasing density functions), and experimental results of training with a genetic algorithm. Also, it is shown that the genetic algorithm with the IS technique attains quasi-optimum training in the sense of minimum error probability (or minimum misclassification probability).

1 Introduction and Preliminaries

Importance Sampling is a modified Monte Carlo technique [1] commonly applied to the performance analysis of radar and communication detectors [2-9]. In communications detectors, the error probability (P_e) is estimated by Importance Sampling (IS) techniques for very low P_e (e.g. $P_e<10^{-5}$). In radar detectors, the very low false-alarm probabilities are also estimated by IS techniques. In paper [10] we have applied IS techniques to neural network detectors for testing performances, where the first part of [10] presents IS as a Monte Carlo technique for computer simulations, and the second part presents false-alarm probability estimations of neural detectors (in the testing phase) applied to radar.

Now, in this paper we propose as a novelty the application of IS techniques in the training phase of neural network detectors. For this purpose, we have to modify adequately the objective functions of our neural networks, as we shall explain in Section 2. Some computer results are presented in Section 3, and conclusions are summarized in Section 4.

Throughout the paper, we shall refer to Fig. 1, where $\mathbf{x}= (x_1, x_2, ..., x_n)$ is the input vector of the R^n-space, $y=g(\mathbf{x})$ is the scalar output, $g(\cdot)$ is a nonlinear system (e. g. a neural network), T_0 is the detection threshold, and $z=u(g(\mathbf{x})-T_0)$ is the detector output, where $u(\cdot)$ is the unit-step function (i.e. $u(t)=1$ if $t>0$ and $u(t)=0$ if $t<0$). We denote $\mathbf{X}=(X_1, X_2, ..., X_n)$ as a random vector and $f_{\mathbf{X}}(\mathbf{x} \mid H_i)$ as the probability density function (pdf) of $\mathbf{X}$ under a hypothesis H_i, $i=1, 0$ (binary hypotheses), where H_0 is the null hypothesis or symbol "0" and H_1 is the alternative hypothesis or symbol "1". $P(H_i)$ is

L. De Raedt and P. Flach (Eds.): ECML 2001, LNAI 2167, pp. 431–441, 2001.

the "a priori" probability of the hypothesis H_i, i= 1, 0, and $P(D_j|H_i)$ is the conditional probability of deciding H_j, j=1, 0, under the true hypothesis H_i, i=1, 0. If $g(\mathbf{x}) > T_0$ (or z=1), the decision is H_1; if $g(\mathbf{x}) < T_0$ (or z=0), the decision is H_0. Finally, $\mathscr{E}\{Z \mid H_i\}$ is the expectation of the random variable Z conditioned by H_i, i=1, 0, and $\mathscr{E}\{g(\mathbf{X})\}$ is the expectation of $g(\cdot)$ with respect to the pdf of $\mathbf{X}$ (i.e. $f_{\mathbf{X}}(\mathbf{x})$).

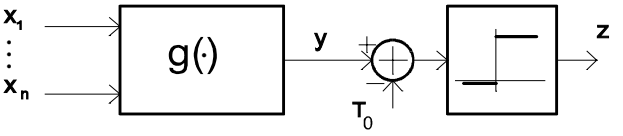

Fig. 1. Binary detector structure

2 Error Probability as Objective Function for Training

To supervise neural network (NN) training, estimations of an objective function (or risk function) have to be performed. The value of this function decreases as the training progresses; then, the number of test patterns required for an accurate estimation has to be increased. Consequently, the training computational cost is unaffordable for very low objective function values, and the use of Importance Sampling (IS) techniques becomes indispensable.

To illustrate the use of IS techniques in the training phase of a NN, let us consider the misclassification probability as an objective function for applications in classifications (or the error probability for detection in communications [11-13]). According to the notation given above, the error probability (P_e) can be expressed as follows

$$P_e = P(H_0)P(D_1 \mid H_0) + P(H_1)P(D_0 \mid H_1) \tag{1}$$

where

$$P(D_1 \mid H_0) = \int_{g(\mathbf{x})>T_0} f_{\mathbf{X}}(\mathbf{x} \mid H_0)d\mathbf{x} \quad \text{and} \quad P(D_0 \mid H_1) = \int_{g(\mathbf{x})<T_0} f_{\mathbf{X}}(\mathbf{x} \mid H_1)d\mathbf{x} \tag{2}$$

Also, in order to save space, let us define

$$h(\mathbf{x}) = P(H_0) f_{\mathbf{X}}(\mathbf{x} \mid H_0)u(g(\mathbf{x}) - T_0) + P(H_1) f_{\mathbf{X}}(\mathbf{x} \mid H_1)u(T_0 - g(\mathbf{x})), \quad \mathbf{x} \in R^n \tag{3}$$

where all items have been defined above.

Now, if we consider a new probability density function (pdf) $f_{\mathbf{X}}^*(\mathbf{x})$, such that $f_{\mathbf{X}}^*(\mathbf{x}) \neq 0$ wherever $h(\mathbf{x})\neq 0$, then from (1), (2) and (3) we have

$$P_e = \int_{R^n} h(\mathbf{x})\,d\mathbf{x} = \int_{R^n} \frac{h(\mathbf{x})}{f_{\mathbf{X}}^*(\mathbf{x})} f_{\mathbf{X}}^*(\mathbf{x})\,d\mathbf{x} = \mathscr{E}^*\left\{\frac{h(\mathbf{X})}{f_{\mathbf{X}}^*(\mathbf{X})}\right\} \tag{4}$$

where $\mathscr{E}^*\{\cdot\}$ means expectation with respect to $f_{\mathbf{X}}^*(\mathbf{x})$ (known as the Importance Sampling pdf).

The last equality in (4) is the key of the Importance Sampling technique. From the statistical inference theory applied to (4), an estimator of P_e is given by

$$P_e^* = \frac{1}{N}\sum_{k=1}^{N}\frac{h(\mathbf{x}_k^*)}{f_{\mathbf{X}}^*(\mathbf{x}_k^*)} \tag{5}$$

where $\mathbf{x}_k^*$, k=1, 2, ..., N, are independent sample vectors whose pdf is $f_{\mathbf{X}}^*(\mathbf{x})$. Estimator P_e^*, given in (5), must be computed in order to perform the neural network training (i.e. to find g(·) for minimum P_e^*).

The mean $\mu_{P_e^*} = \mathscr{E}^*\{P_e^*\} = P_e$ and the variance $\sigma_{P_e^*}^2$ of the estimator P_e^* is given by

$$\sigma_{P_e^*}^2 = \mathscr{E}^*\left\{(P_e^* - \mu_{P_e^*})^2\right\} = \frac{1}{N}\left[\mathscr{E}^*\left\{\left(\frac{h(\mathbf{X}_k^*)}{f_{\mathbf{X}}^*(\mathbf{X}_k^*)}\right)^2\right\} - P_e^2\right] \tag{6}$$

Then, P_e^* is an unbiased and consistent estimator of P_e (i.e. $P_e^* \to P_e$ as $N \to \infty$).

The estimation error $\varepsilon_{P_e^*}$ of the estimator P_e^* is defined by

$$\varepsilon_{P_e^*} = \frac{\sigma_{P_e^*}}{\mu_{P_e^*}} = \sqrt{\frac{1}{N}\left(\frac{\mathscr{E}^*\left\{\left(\frac{h(\mathbf{X}_k^*)}{f_{\mathbf{X}}^*(\mathbf{X}_k^*)}\right)^2\right\}}{\left(P_e\right)^2} - 1\right)} \tag{7}$$

Because the variance is not a negative number, from (6) we have

$$\mathscr{E}^*\left\{\left(\frac{h(\mathbf{X}_k^*)}{f_{\mathbf{X}}^*(\mathbf{X}_k^*)}\right)^2\right\} \geq P_e^2 \tag{8}$$

The equality case in (8) is satisfied if

$$f_{\mathbf{X}}^*(\mathbf{x}) = \frac{1}{P_e}h(\mathbf{x}), \quad \mathbf{x} \in R^n \tag{9}$$

that it can be proved by taking (9) into (8); then, the estimator variance in (6) is zero for any value of N, i.e. only one sample vector (N=1) is required for estimating P_e without error (estimation error (7) is zero). Expression (9) is the unconstrained optimal solution for $f_{\mathbf{X}}^*(\mathbf{x})$. Note that $h(\mathbf{x}) \geq 0$, $\mathbf{x} \in R^n$ (from definition (3)), then expression (9) is a pdf because of (4).

By taking (3) into (9), the unconstrained optimal solution for $f_{\mathbf{X}}^*(\mathbf{x})$ can be expressed for future references as follows

$$f_{\mathbf{X}}^{*}(\mathbf{x}) = \frac{1}{P_e}\left[P(H_0) f_{\mathbf{X}}(\mathbf{x} \mid H_0) u(g(\mathbf{x}) - T_0) + P(H_1) f_{\mathbf{X}}(\mathbf{x} \mid H_1) u(T_0 - g(\mathbf{x}))\right], \mathbf{x} \in R^n \quad (10)$$

The optimal solution for $f_{\mathbf{X}}^{*}(\mathbf{x})$ given in (10) is not realistic, because P_e is not known "a priori" (it has to be estimated by (5)). Furthermore, in the training phase, $g(\cdot)$ is changing from one iteration to the other.

A suboptimal solution for $f_{\mathbf{X}}^{*}(\mathbf{x})$ is obtained in a way similar to that done in [10]. Usually $f_{\mathbf{X}}(\mathbf{x} \mid H_i)$, $i = 1, 0$, depends on a parameter θ (e.g. the signal-to-noise ratio) [12,13] and we can write $f_{\mathbf{X}}(\mathbf{x}; \theta \mid H_i)$, $i = 1, 0$; if θ=0 there is only noise (both pdf's are identical to the noise pdf); if θ is too large, both hypotheses are highly separate (and corresponding to very low error probability). Now, we consider the following (suboptimal) density function as the Importance Sampling pdf

$$f_{\mathbf{X}}^{*}(\mathbf{x}) = P(H_0) f_{\mathbf{X}}(\mathbf{x}; \theta^{*} \mid H_0) + P(H_1) f_{\mathbf{X}}(\mathbf{x}; \theta^{*} \mid H_1), \quad \mathbf{x} \in R^n \quad (11)$$

where θ^{*} is the θ-value that minimizes the variance (6) of the estimator P_e^{*}. Note that (11) satisfies the necessary condition for the unbiasedness of P_e^{*}, i.e. $f_{\mathbf{X}}^{*}(\mathbf{x}) \neq 0$ as $h(\mathbf{x}) \neq 0$, $\mathbf{x} \in R^n$, where $h(\mathbf{x})$ is given by (3).

The optimal θ^{*}-value is obtained experimentally by computing an estimator of (6). An estimator of (6) is given by

$$\hat{\sigma}^2_{P_e^{*}} = \frac{1}{N}\left[\frac{1}{N}\sum_{k=1}^{N}\left(\frac{h(\mathbf{x}_k^{*})}{f_{\mathbf{X}}^{*}(\mathbf{x}_k^{*})}\right)^2 - \left(P_e^{*}\right)^2\right] \quad (12)$$

where P_e^{*} is given by (5) and $h(\mathbf{x})$ is given by (3). Then, θ^{*} can be estimated experimentally and corresponds to the θ-value that minimizes (12). However, a better statistical parameter for this purpose is the relative error of P_e^{*} given by (7). An estimator of the relative error (7) (denoted as $\hat{\varepsilon}_{P_e^{*}}$) is

$$\hat{\varepsilon}_{P_e^{*}} = \frac{\hat{\sigma}_{P_e^{*}}}{\hat{\mu}_{P_e^{*}}} = \sqrt{\frac{1}{N}\left(\frac{\frac{1}{N}\sum_{k=1}^{N}\left(\frac{h(\mathbf{x}_k^{*})}{f_{\mathbf{X}}^{*}(\mathbf{x}_k^{*})}\right)^2}{\left(P_e^{*}\right)^2} - 1\right)} \quad (13)$$

where $h(\mathbf{x})$, $f_{\mathbf{X}}^{*}(\mathbf{x})$ and P_e^{*} are given by (3), (11) and (5), respectively.

Expressions (5) and (13) are computed at the same time for a given θ (fixed during the training), N (that can be adjusted in each iteration), and $g(\cdot)$ which changes as the training progresses. At the end of the training, P_e^{*} and $\hat{\varepsilon}_{P_e^{*}}$ are good estimations of P_e and $\varepsilon_{P_e^{*}}$, respectively. After some computer runs with different θ-values, the optimal θ^{*} of minimum $\hat{\varepsilon}_{P_e^{*}}$ can be obtained by inspection.

In a general application, where parameter θ can be a vector, the minimization of (13) have to be performed by an adequate optimization algorithm like stochastic gradient descent algorithms [4,5] or, alternatively, by genetic algorithms. Nowadays, this subject is under consideration by our research group.

Finally, we have to point out that (11) is a suboptimal solution of $f_{\mathbf{X}}^{*}(\mathbf{x})$ (optimum expressed by (10)), if the neural network is already trained (or quasi-trained), as is stated and fulfilled in [10], because Importance Sampling was applied there for testing performances. Nevertheless, (11) can be used also in the first iterations of the training because the error probability (P_e) is high and the error probability estimation (P_e^*) is also high (although inaccurate). In the last iterations of the training (network quasi-trained), both P_e and P_e^* are low (or very low) and close each other. These facts were also tested by means of computer simulations.

3 Computer Simulations

In order to show how Importance Sampling works in neural detectors, consider a detection of binary symbols in Gaussian noise. The hypotheses are $H_1 : \mathbf{x} = \boldsymbol{\eta} + \mathbf{a}$ and $H_0 : \mathbf{x} = \boldsymbol{\eta} - \mathbf{a}$, where $\mathbf{a} = (a_1, a_2, ..., a_n)$, $a_i = \mu$, $i = 1, 2, ..., n$ and μ is a real constant (for simulations μ=2), $\boldsymbol{\eta} = (\eta_1, \eta_2, ..., \eta_n)$ is a Gaussian noise vector of independent and identically distributed zero-mean components of unit variance. Then, the pdf's under each hypothesis are normal distributed with means **a** and **–a**, respectively, i.e.

$$\begin{aligned} f_{\mathbf{X}}(\mathbf{x} \mid H_1) &= (2\pi)^{-n/2} \exp\left(-\frac{1}{2} \sum_{i=1}^{n} (x_i - \mu)^2 \right) \\ f_{\mathbf{X}}(\mathbf{x} \mid H_0) &= (2\pi)^{-n/2} \exp\left(-\frac{1}{2} \sum_{i=1}^{n} (x_i + \mu)^2 \right) \end{aligned} \tag{14}$$

Also, we suppose for simulations that $P(H_1) = P(H_0) = 1/2$ (the symbols are equally likely).

Referring to Fig. 1, a Multi-Layer Perceptron (MLP) is the NN used as nonlinear system $g(\mathbf{x})$. The parameters for the MLP are 5×5×1 (i.e. number of input nodes: n=5, number of hidden-layer nodes: 5, number of outputs: 1) and the threshold T_0=0.5. A genetic algorithm [14] is used for training the MLP. Genetic algorithms for optimization are very close to Monte Carlo techniques, so Importance Sampling is well tailored for training neural networks by genetic algorithms. Computer programs for implementing both genetic algorithm and Importance Sampling are independent, and a main program calls each one. Although our genetic algorithm (with elitism) is not the subject of this paper (in fact, here it is only considered as a tool), we give the parameters used in our training. These are the following: number of MLP's in genetic set: 20; mutation probability: 0.1; crossover probability: 0.1; fitness function: $-\log(P_e^*)$; number of iterations (generations): 100. The number of patterns (input sample vectors) for estimating P_e^* by IS technique is N ($10^2 < N < 10^3$); in fact, it is

required N>200 for efficient training. Finally, for this application, back-propagation training does not work well because the objective function (5) is discontinuous (due to the unit-step function in (3)).

In this example, after numerical computations of (13) with $\theta=\mu$, we obtain the solution $\theta^*=0$ which minimizes expression (13); consequently, from (11) we have that the noise pdf is the suboptimal solution for the IS pdf, i.e.

$$f_{\mathbf{X}}^*(\mathbf{x}) = (2\pi)^{-n/2} \exp\left(-\frac{1}{2}\sum_{i=1}^{n} x_i^2 \right) \tag{15}$$

In Fig. 2, we show the results of the error probability estimations (P_e^*) versus iteration number. Also, for $\theta^*=0$ in (11), we have (15) as our IS pdf to generate patterns for training. We have used a number of patterns N=200 in the first iterations of the training process, and $N=10^3$ in the last ones (N increases from 200 to 10^3, following a cubic parabola as the training progresses). So, N increases slowly at the beginning of the training, and very fast at the end. In Fig. 3, we present the relative error $\hat{\varepsilon}_{P_e^*}$ computed from (13) at the same time that P_e^* is computed from (5). In the first iterations of the training, both P_e^* and $\hat{\varepsilon}_{P_e^*}$ are high, and inaccurate estimations of P_e and $\varepsilon_{P_e^*}$, respectively, because of two reasons. The first reason is the low number of vector samples (N=200) in the computations of P_e^* and $\hat{\varepsilon}_{P_e^*}$; the second reason is that $f_{\mathbf{X}}^*(\mathbf{x})$ (corresponding to $\theta^*=0$) departs from the optimum one. In the last iterations of the training, $N=10^3$ and $f_{\mathbf{X}}^*(\mathbf{x})$ corresponds to the optimum ($\theta^*=0$), then both P_e^* and $\hat{\varepsilon}_{P_e^*}$ are highly accurate ($P_e^* \approx 4\cdot 10^{-6}$ with an error $\hat{\varepsilon}_{P_e^*} \approx 5\%$ from Figs. 2 and 3, respectively, and the iteration number equals 100).

We can see a quick convergence of the training (less than 60 iterations) and the quasi-optimality of the resultant neural detector with $P_e^* \approx 4\cdot 10^{-6}$, which is close to the optimal Bayes' detector (linear detector) with $(P_e)_{\text{Bayes}} = 3.87\cdot 10^{-6}$. On the other hand, if we use a standard Monte Carlo simulation for estimating P_e (denoted as $\hat{P}_e$), we need more than $N=10^7$ patterns (compare with $N=10^3$ of the Importance Sampling). For the case of a Monte-Carlo simulation with $N=10^7$ we have obtained an error probability estimation $\hat{P}_e = 4.8\cdot 10^{-6} \approx P_e^*$ (within a precision of 15%).

In Figs. 4 and 5 we have considered the same conditions and parameters to those of Figs. 2 and 3, except for the number of patterns (N). Now, $N=10^3$ for each iteration number (the same number of patterns in the first iterations and in the last ones). As it can be seen, Figs. 2 and 3 are noisier than Figs. 4 and 5, respectively; however, Figs. 2 and 4 converge to the same value ($\approx 4\cdot 10^{-6}$) with the same precision (5% of precision from Figs. 3 and 5), whenever the 80th iteration is reached. On the other hand, the time required for Figs. 2 and 3 was 3 minutes in a Pentium III-PC at 800 Mhz, and 9 minutes for Figs. 4 and 5 (programs were written in Matlab). Consequently, it is better to use: $200<N<10^3$ in an adjustable way, depending on the training iteration number.

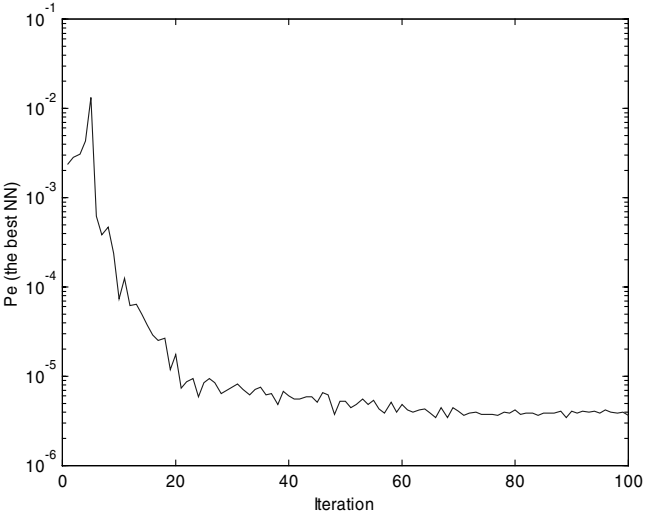

Fig. 2. Error Probability (*Pe*) versus iteration number for the neural detector training, using the Importance Sampling technique. An MLP of 5×5×1 nodes, and a genetic algorithm for training with 200 patterns (input sample vectors) in the first iterations and 1000 patterns in the last ones. The IS parameter θ^{*}=0 (optimum for the IS pdf).

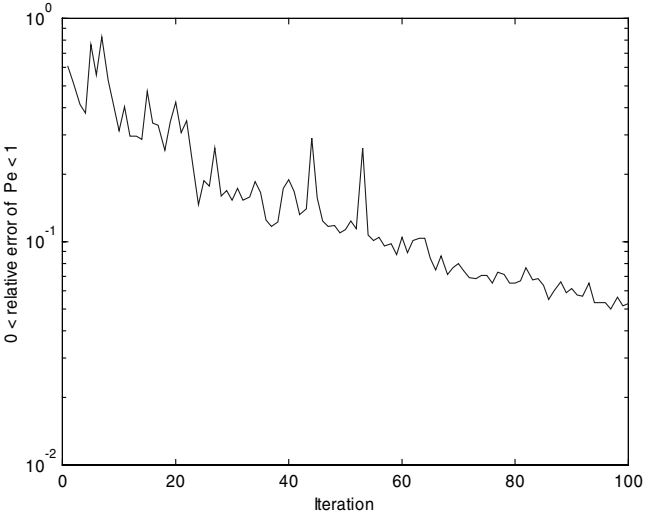

Fig. 3. Relative Error ($\hat{\varepsilon}_{P_e}$) of *Pe* (corresponding to Fig. 2) versus iteration number for the neural detector training, using the Importance Sampling technique. An MLP of 5×5×1 nodes, and a genetic algorithm for training under the conditions of Fig. 2.

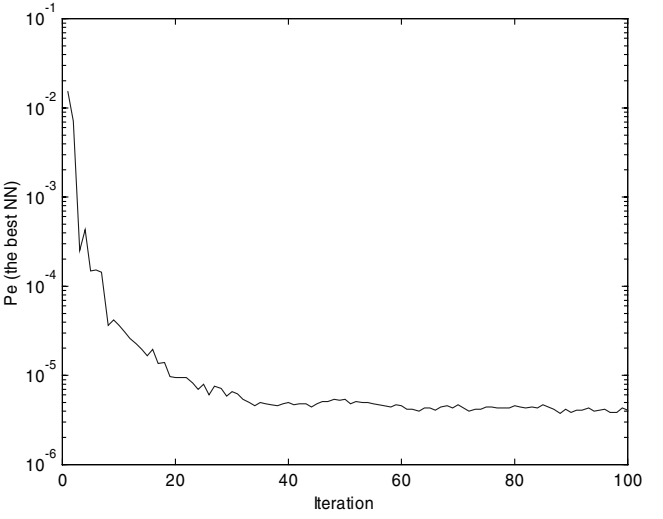

Fig. 4. Error Probability (*Pe*) versus iteration number for the neural detector training, using the Importance Sampling technique. An MLP of 5×5×1 nodes, and a genetic algorithm for training with 1000 patterns (input sample vectors) in each iteration. The IS parameter θ^*=0 (optimum).

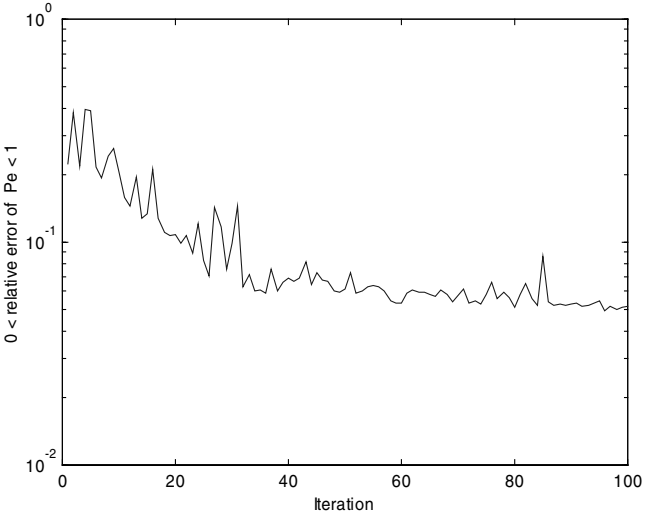

Fig. 5. Relative Error ($\hat{\varepsilon}_{P_e}$) of *Pe* (corresponding to Fig. 4) versus iteration number for the neural detector training, using the Importance Sampling technique. An MLP of 5×5×1 nodes, and a genetic algorithm for training under the conditions of Fig. 4.

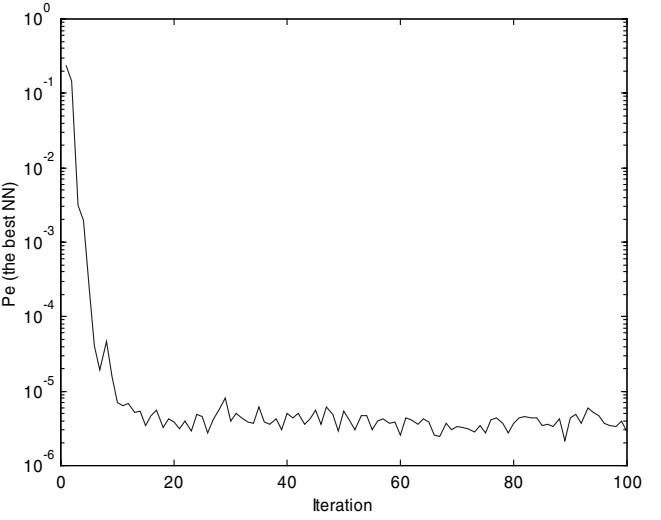

Fig. 6. Error Probability (*Pe*) versus iteration number for the neural detector training, using the Importance Sampling technique. An MLP of 5×5×1 nodes, and a genetic algorithm for training with 1000 patterns (input sample vectors) in each iteration. The IS parameter θ=1 (which is non-optimum for IS pdf).

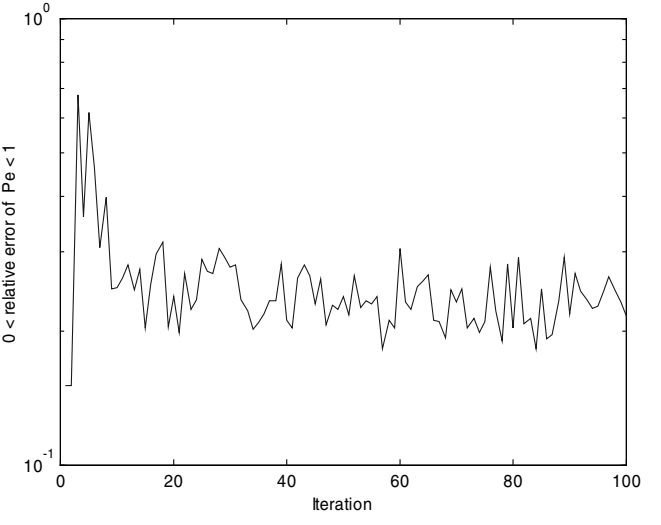

Fig. 7. Relative Error ($\hat{\varepsilon}_{P_e}$) of *Pe* (corresponding to Fig. 6) versus iteration number for the neural detector training, using the Importance Sampling technique. An MLP of 5×5×1 nodes, and a genetic algorithm for training under the conditions of Fig. 6.

In Figs. 6 and 7 we have considered the same conditions and parameters to those of Figs. 4 and 5, except for the IS parameter θ. Now, θ=1 is non-optimum for the IS pdf given by (11), because (13) is high ($\hat{\varepsilon}_{P_e^*} \approx 25\%$ in Fig. 7, and compare with Fig. 5).

Similar conclusions can be obtained for other MLP structures (e.g. 8×8×1 or 8×4×1, 10×5×1, etc.), trained under the conditions given above. For example, for the case 8×8×1, we have $P_e^* \approx 9 \cdot 10^{-9}$; for the case 8×4×1, we have $P_e^* \approx 10^{-8}$; then both are quasi-optimum, because $(P_e)_{\text{Bayes}} = 7.7 \cdot 10^{-9}$ (the optimal Bayes detector). Finally, another issue is the analysis of the robustness of these neural detectors, i.e. testing our neural detectors against hypotheses slightly different from the training hypotheses (analysis of departures from the assumptions).

4 Conclusions

Importance Sampling (IS) techniques have been presented in order to drastically accelerate "low error probability" estimations, needed to supervise the Neural Network training in detection applications.

Adequate modifications on the error probability objective function were realized in order to use standard training algorithms for neural detector training. This approach is useful in communications systems, where very low error probabilities are concerned. Also, generalizations to other types of objective functions are straightforward.

The suboptimal IS probability density function (biasing density function) corresponding to the trained conditions is useful for all the training process. From empirical examples, we have shown that the number of 10^7 (or more) input sample vectors (patterns) per iteration, required by standard Monte Carlo techniques, is reduced to 10^3 (or even less) patterns per iteration with the IS method. This reduction is independent of the training algorithm type. Note that back-propagation algorithm requires continuity of the objective function, so that its use is problematic in our application (due to the unit-step function appears in our objective function).

Finally, further work is required to consider adaptive IS probability density functions in the training (a technique that consists of improving the IS probability density function as the training progresses). Also, the training of neural detectors under Neyman-Pearson's criterion and Importance Sampling is now under consideration in our research group.

References

1. Thompson, S.K.: Sampling. John Wiley & Sons (Wiley-Interscience), New York (1992).
2. Smith, P.J., Shafi, M., and Gao, H.: Quick Simulation: A Review of Importance Sampling Techniques in Communications Systems. IEEE J. Select. Areas Commun., 1997, 15, (4), pp. 597-613.
3. Chen, J.-C., Lu, D., Sadowsky, J.S., and Yao, K.: On Importance Sampling in Digital Communications—Part I: Fundamentals. IEEE J. Select. Areas Commun., 1993, 11, (3), pp. 289-299.

4. Al-Qaq, W.A., Devetsikiotis, M., and Townsend, J.K.: Stochastic Gradient Optimization of Importance Sampling for the Efficient Simulation of Digital Communication Systems. IEEE Trans. Commun., 1995, 43, (12), pp. 2975-2985.
5. Al-Qaq, W.A., and Townsend, J.K.: A Stochastic Importance Sampling Methodology for the Efficient Simulation of Adaptive Systems in Frequency Nonselective Rayleigh Fading Channels. IEEE J. Select. Areas Commun., 1997, 15, (4), pp. 614-625.
6. Gerlach, K.: New Results in Importance Sampling. IEEE Trans. Aerosp. Electron. Syst., 1999, 35, (3), pp. 917-925.
7. Orsak, G.C.: A Note on Estimating False Alarm Rates via Importance Sampling. IEEE Trans. Commun., 1993, 41, (9), pp. 1275-1277.
8. Orsak, G.C., and Aazhang, B.: Constrained Solutions in Importance Sampling via Robust Statistics. IEEE Trans. Inform. Theory, 1991, 37, (2), pp. 307-316.
9. Sadowsky, J.S., and Bucklew, J.A.: On Large Deviation Theory and Asymptotically Efficient Monte Carlo Estimation. IEEE Trans. Inform. Theory, 1990, 36, (3), pp. 579-588.
10. Sanz-González, J.L., and Andina, D.: Performance Analysis of Neural Network Detectors by Importance Sampling Techniques. Neural Processing Letters, 1999, 9, (3), pp. 257-269.
11. Andina, D., Sanz-González, J.L., and Jiménez-Pajares, A.: A Comparison of Criterion Functions for a Neural Network Applied to Binary Detection. Proc. of IEEE Int. Conf. on Neural Networks (ICNN'95), 1995, Vol. 1, pp. 329-333.
12. Poor, H.V.: An Introduction to Signal Detection and Estimation, (Second Edition) Springer-Verlag, Berlin (1994).
13. Van Trees, H.L.: Detection, Estimation, and Modulation Theory. Part. I. John Wiley & Sons, New York (1968).
14. Seijas, J. and Sanz-González, J.L.: Basic-Evolutive Algorithms for Neural Networks Architecture Configuration and Training. Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'95), 1995, Vol. 1, pp. 125-130.