Cost Functions to Estimate *A Posteriori* Probabilities in Multiclass Problems

Jesús Cid-Sueiro, *Member, IEEE,* Juan Ignacio Arribas, Sebastián Urbán-Muñoz, and Aníbal R. Figueiras-Vidal, *Senior Member, IEEE*

***Abstract*—The problem of designing cost functions to estimate *a posteriori* probabilities in multiclass problems is addressed in this paper. We establish necessary and sufficient conditions that these costs must satisfy in one-class one-output networks whose outputs are consistent with probability laws. We focus our attention on a particular subset of the corresponding cost functions; those which verify two usually interesting properties: symmetry and separability (well-known cost functions, such as the quadratic cost or the cross entropy are particular cases in this subset). Finally, we present a universal stochastic gradient learning rule for single-layer networks, in the sense of minimizing a general version of these cost functions for a wide family of nonlinear activation functions.**

***Index Terms*—Neural networks, pattern classification, probability estimation.**

I. Introduction

DECISION, classification, and detection, are old words that have been recently adopted for use in technical theories. For example, the Spanish verb "decidir" (to decide) appeared in 1569, originating from the Latin word "decidere" (and this word from "caedere," which means to cut!). To say yes or no, this or that, that there is or there is not, respectively, are the answers to the corresponding questions associated with decision, classification, and detection problems. It is obvious that each question can be reformulated in the form of another, but it is also evident that the subjective implications are not the same.

Fortunately, the Bayesian formulation of these problems can be adapted to these general differences, as well as to other of less basic relevance: the use of decision cost functions serve to balance all them. Once the decision costs are established, data likelihoods and *a priori* probabilities or *a posteriori* probabilities of the hypotheses are enough to obtain optimal results [1], [2].

In practice, with the exception of cases in which the "mechanics" of the problem are known (as in many transmission problems), likelihoods or *a posteriori* probabilities must be estimated. In particular, neural networks can be used to estimate *a posteriori* probabilities by means of a supervised training [3]–[6].

The estimation of *a posteriori* probabilities is not required in order to make a decision. Indeed, Vapnik [7] remarks that to include this estimation increases the complexity of the process. This fact was the origin of Rosenblatt's Perceptron Rule, along with its many variants which deal with the lack of convergence in nonseparable situations [8]. Even structural and training generalizations, such as the decision-based neural networks [9] have difficulties; to solve them is an NP-complete problem [10] and only suboptimal procedures can be applied in practice.

Since perceptron-rule-based algorithms have these drawbacks and generalize poorly, other authors have followed alternative routes for minimum error decision: Telfer and Szu, for example, use Minkowski's L_1-norm minimization and a very steep sigmoidal activation output [11], [12]. At the same time, other objective functions have appeared to address robustness, training speed, decision performance, etc. Reference [13] presents a very complete overview, and [14] proposes using Csiszár measures [15] in several forms, following the idea of minimizing divergences introduced by Hopfield [16] and Hinton [17].

However, in many cases, proposing cost functions for strict decision purposes is not enough [e.g., in (auxiliary) automatic diagnostic systems for clinical applications, in financial problems, or when decision fusion will be a final step]. In these cases, *a posteriori* probability estimates are natural quantities to be considered.

Surprisingly, the if and only if conditions for a cost function providing estimates of the *a posteriori* probabilities have only been established for binary problems, although there have been discussions addressing this property for specific cost functions in multiple hypotheses situations. In this paper, we focus on the general M-ary case, assuming that, as usual, cost function $C(\mathbf{y}, \mathbf{d})$ depends only on output $\mathbf{y}$ and desired target $\mathbf{d}$. After finding the corresponding iff condition, we derive some theoretical and practical implications about the characteristics of the corresponding estimates.

It should be noted that the analysis we carry out implicitly assumes that the architecture under consideration is general enough to have the capability of approaching the desired *a posteriori* probabilities (and even that training leads to the global optimum). Of course, this is not true in all practical situations; however we show that if the network is not general enough, the proposed cost functions provide the nearest *a posteriori* probability estimates according to some divergence measure.

Manuscript received September 12, 1997; revised January 21, 1999. This work was supported in part by CICYT under Grants TIC96-0500-C10-03 and -09.

J. Cid-Sueiro and J. I. Arribas are with the Departamento de Teoría de la Señal y Comunicaciones e Ing. Telemática, ETSIT, Universidad de Valladolid, Campus Miguel Delibes s/n, 47011 Valladolid, Spain.

A. R. Figueiras-Vidal and S. Urbán-Muñoz are with the Dpto. de Tecnologías de las Comunicaciones, EPS/Universidad Carlos III de Madrid, 28911 Leganés-Madrid, Spain.

Publisher Item Identifier S 1045-9227(99)03832-1.

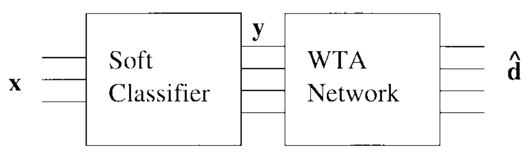

Fig. 1. Soft decision multioutput classifier.

The organization of this paper is as follows. Section II states the probability estimation problem. In Section III we show general conditions for the cost functions that provide estimates of the *a posteriori* class probabilities. Section IV studies the convexity of the proposed functions, and Section V shows that the proposed cost functions minimize a divergence measure between probabilities. Section VI explores the application of these theoretical results to single layer networks. A simulation example and a general discussion of the advantages of the proposed cost functions can be found in Section VII, and Section VIII summarizes the main conclusions and suggests some lines for future research.

II. Problem Formulation

A multiple classification problem can be stated in the following manner. Some system produces a class index c according to probabilities $P(c)$, $c = 1, \cdots, L$; then, an observation vector $\mathbf{x}$ is generated according to probability density function $f(\mathbf{x}|c)$. After observing $\mathbf{x}$, the classifier system has to decide which class generated it.

A soft-decision-based classifier makes decisions in two steps (see Fig. 1). First, it computes soft-decision vector $\mathbf{y}$ with components

$$y_i = h_i(\mathbf{x}, \mathbf{w}) \tag{1}$$

where $\mathbf{w}$ is a vector of network parameters. We assume that outputs y_i of activation functions h_i verify

$$\sum_{i=1}^{L} y_i = 1 \tag{2}$$

(e.g., using the softmax nonlinearity). Second, the classifier makes a hard decision [a winner-take-all (WTA) network is assumed] so that the components of output decision vector $\hat{\mathbf{d}}$ are given by

$$\hat{d}_i = \delta_{j-i} \tag{3}$$

where δ is the Kronecker delta and

$$j = \arg\max\{y_i, i = 1, \cdots, L\}. \tag{4}$$

A decision error occurs when $\hat{\mathbf{d}}$ is not equal to target vector $\mathbf{d}$ with components $d_i = \delta_{c-i}$, where c is the class of the observation. The error probability is minimum for class

$$j_0 = \arg\max\{P(i|\mathbf{x}), \quad i = 1, \cdots, L\}. \tag{5}$$

Besides making optimal decisions ($j = j_0$), we are interested in the class of networks whose soft decisions are equal to the *a posteriori* class probabilities ($y_i = P(i|\mathbf{x})$). The corresponding cost function must be strict sense Bayesian (SSB), which we define as follows.

Definition 1: Let $S = \{\mathbf{y} | 0 \leq y_i \leq 1, \sum_{i=1}^{L} y_i = 1\}$. A cost function $C(\mathbf{y}, \mathbf{d})$ is said to be SSB if $E\{C(\mathbf{y}, \mathbf{d})|\mathbf{x}\}$ has a unique minimum in S when the outputs are the *a posteriori* probabilities of the classes, i.e.,

$$y_i = P(d_i = 1|\mathbf{x}), \qquad i = 1, \cdots, L. \tag{6}$$

Since

$$E\{C(\mathbf{y}, \mathbf{d})\} = \int E\{C(\mathbf{y}, \mathbf{d})|\mathbf{x}\} f_x(\mathbf{x})\, d\mathbf{x} \tag{7}$$

and $f_x(\mathbf{x})$ is always nonnegative, minimizing $E\{C(\mathbf{y}, \mathbf{d})|\mathbf{x}\}$ for every input vector $\mathbf{x}$ is equivalent to minimizing $E\{C(\mathbf{y}, \mathbf{d})\}$. In the following, we will use the notation

$$E_c(\mathbf{y}, \mathbf{p}) = E\{C(\mathbf{y}, \mathbf{d})|\mathbf{x}\} \tag{8}$$

where $\mathbf{p}$ is a vector with components

$$p_i = P(d_i = 1|\mathbf{x}), \qquad i = 1, \cdots, L. \tag{9}$$

Although most of this paper is concerned with multioutput classifiers, we will also consider the problem of binary classifiers with a single output. The definition for an SSB cost function in this case is as follows.

Definition 2: Let $S' = [0, 1]$. Consider a single-output binary classifier with soft decision y, $0 \leq y \leq 1$ and hard decision d equal to the integer nearest to y. A cost function $C(y, d)$ is said to be SSB if $E\{C(y, d)|\mathbf{x}\}$ has a unique minimum in S' when the outputs are the *a posteriori* probabilities of class 1, i.e.,

$$y_{\min} = \arg_y \min E\{C(y, d)|\mathbf{x}\} = P(d = 1|\mathbf{x}). \tag{10}$$

According to the above definitions, an SSB detector requires an SSB cost function. For instance, the well-known cross-entropy (logarithmic cost) function given by

$$C(\mathbf{y}, \mathbf{d}) = \sum_{i=1}^{L} d_i \log(y_i) \tag{11}$$

is SSB [19], [20]. Not every cost function is SSB, for instance, an L_p norm is SSB only for $p = 2$ [4], [21].

Besides a learning algorithm, an SSB classifier requires a network structure able to construct class probabilities and optimal decisions. In the following, we will assume that the network satisfies the complexity requirements of the data distribution for any case, since our interest is not centered on any specific architecture. Section V-B discusses the use of SSB cost functions for insufficient networks.

General conditions for a cost function providing *a posteriori* probability estimates have been studied by Hampshire and Pearlmutter [3] in the binary case; for the same case, Miller *et al.* [6] extended their work, showing that any SSB cost function can be expressed in the form

$$C(y, d) = \int_{K}^{y} g(\alpha)(\alpha - d)\, d\alpha \tag{12}$$

where $g(\alpha) \geq 0$ is a positive function, and K is an arbitrary constant. In practice, the selection of any cost in the family can be done by parameterizing $g(y)$ and giving values to the parameters, for instance by using the Taylor series expansion of $g(y)$ as suggested in [22].

The multiclass problem can be addressed using different network configurations:

1) single output networks where classes are denoted with scalars d_i, $i = 1, \cdots, L$, and L is the number of classes;
2) a network with one output per class;
3) a network with one output per class and outputs constrained by

$$\sum_{i=1}^{L} y_i = 1. \tag{13}$$

The first case is also addressed in [6], where it is shown that

$$y = E\{d|\mathbf{x}\} = \sum_{i=1}^{L} P(d_i|\mathbf{x})\, d_i \tag{14}$$

where d is the target, is the unique minimum of cost function $C(y, d)$ iff it can be expressed as (12) [i.e., the SSB costs given by (12) provide estimates of the conditional mean value of the target].

Note that if the output is given by (14), the extreme values of d are penalized. For instance, if $d_1 < d_2 < \cdots < d_L$, all terms $P(d_i|\mathbf{x})d_i$ in (14) with $i > 1$ “push” the output in the same direction away from d_1 and, thus, the network decision will be d_1 only if $P(d_1|\mathbf{x})$ is clearly dominant over the other probabilities.

As shown in [3], the results obtained for the two-class problem can be easily extended to the second case: any cost function given by a linear combination of costs given by (12) is SSB. The usual configurations of the multilayer-perceptron are examples of this kind of network.

Since the goal is to estimate probabilities, the use of networks whose L outputs are constrained by (13) is more adequate. A multioutput network with a softmax output activation function [23], [24] is an example. We consider in this paper networks satisfying this probability law.

III. General SSB Cost Functions

This section provides theoretical results on the conditions that SSB cost functions must satisfy in multioutput networks. A general formula for an SSB cost function is given by the following theorem.

Theorem 1: A cost function $C(\mathbf{y}, \mathbf{d})$ is SSB iff it can be written in the form

$$C(\mathbf{y}, \mathbf{d}) = v(\mathbf{y}) + \sum_{i=1}^{L} \frac{\partial v}{\partial y_i}(d_i - y_i) \tag{15}$$

where $v(\mathbf{y})$ is any convex function in S which does not depend on d.

Proof: See Appendix A. ■

The previous result shows that any SSB cost is completely specified by a convex and multidimensional function $v(\mathbf{y})$. Note that the only constraint imposed on C is that it must be a function of $\mathbf{y}$ and $\mathbf{d}$, that is, the entire influence of the network parameters on the cost function is given by $\mathbf{y}$. Now, we define the concepts of separability, which simplifies the characterization of the SSB cost functions, and symmetry, which is usually required in practice.

A. Separable Cost Functions

We focus here on the special and sometimes desirable case of SSB cost functions with uncoupled components. We present the following definition and theorem.

Definition 3—Separable Cost Function: A cost function is *separable* if it can be represented in the form

$$C(\mathbf{y}, \mathbf{d}) = \sum_{i=1}^{L} c_i(y_i, d_i). \tag{16}$$

Theorem 2: $C(\mathbf{y}, \mathbf{d})$ is a separable SSB cost function iff it can be written in the form

$$C(\mathbf{y}, \mathbf{d}) = \sum_{i=1}^{L} \int_{d_i}^{y_i} g_i(\alpha)(\alpha - d_i)\, d\alpha + r(\mathbf{d}) \tag{17}$$

where $0 \le \alpha \le 1$, g_i are positive functions $(g_i(\alpha) > 0)$ and $r(\mathbf{d})$ is an arbitrary function which does not depend on $\mathbf{y}$.

Proof: See Appendix B. ■

The selection of $r(\mathbf{d})$ is irrelevant because it does not depend on the network weights. Thus, separable SSB cost functions are described by L scalar functions $g_i(\alpha)$. In order to specify a particular cost in the subset, the parametric technique described in [22] for the binary case can be easily applied to the multiclass problem. Note also that, when using gradient algorithms, the integral functions do not have to be computed because learning rules are based on the cost derivatives.

B. Symmetric Costs

In most applications, it is expected that the classifier performance should not depend on the order of the input vector components. The SSB cost functions derived before do not satisfy, in general, this kind of symmetry. For this case we present the following definition and theorem.

Definition 4—Symmetric Cost Function: A separable cost function is *permutationally symmetric*, or simply *symmetric*, if it is invariant when the components of $\mathbf{y}$ and $\mathbf{d}$ are reordered simultaneously

$$C(\mathbf{y}, \mathbf{d}) = C(Q(\mathbf{y}), Q(\mathbf{d})) \tag{18}$$

where $Q(\mathbf{y})$ is an arbitrary permutation of the components of $\mathbf{y}$.

Theorem 3: A symmetric and separable cost function $C(\mathbf{y}, \mathbf{d})$ is SSB iff it can be written in the form

$$C(\mathbf{y}, \mathbf{d}) = \sum_{i=1}^{L} \int_{d_i}^{y_i} g(\alpha)(\alpha - d_i)\, d\alpha + r(\mathbf{d}) \tag{19}$$

where $g(\alpha)$ is any positive function $(g(\alpha) > 0,\ 0 \le y \le 1)$ which does not depend on $\mathbf{d}$, and $r(\mathbf{d})$ is an arbitrary function which does not depend on $\mathbf{y}$.

Proof: See Appendix C. ■

The symmetry property simplifies the design of the cost function to the determination of a unique scalar function $g(\alpha)$. Equation (19) is a direct extension of (12) to the multioutput case.

It is easy to verify that many classical cost functions satisfy the above conditions. For example, for $g(\alpha) = 2$ and $r(\mathbf{d}) = 0$, we have the conventional quadratic cost

$$C(\mathbf{y}, \mathbf{d}) = \sum_{i=1}^{L} (d_i - y_i)^2 = \|\mathbf{y} - \mathbf{d}\|^2 \tag{20}$$

for $g(\alpha) = 1/\alpha$ and $r(\mathbf{d}) = 0$, we get

$$C(\mathbf{y}, \mathbf{d}) = \sum_{i=1}^{L} d_i \log y_i \tag{21}$$

which is the cross-entropy between probability distribution $\mathbf{y}$ and degenerate distribution $\mathbf{d}$.

IV. Concavity

Although we have demonstrated that the global and unique minimum of the proposed cost functions is found at the class probabilities, concavity of mean cost E_c is not guaranteed. It is a desirable property, because it ensures that, the farther the output vector $\mathbf{y}$ is from $\mathbf{p}$, the higher the gradient of the mean cost is and the higher (on the average) the weight change in the learning rules. Thus, additional constraints on $v(\mathbf{y})$ are needed to guarantee concavity of C as a function of $\mathbf{y}$.

General concavity conditions are easy to find when the SSB cost is separable, as described in the following theorem.

Theorem 4: The mean value of a separable SSB cost is a concave function of $\mathbf{y}$ iff $g_j(\alpha)(1-\alpha)$, $j = 1, \cdots, L$, is an increasing function of α and $g_j(\alpha)\alpha$, $j = 1, \cdots, L$, is a decreasing function of α.

Proof: From (17), we get

$$\frac{\partial E_c}{\partial y_j} = -\, g_j(y_j)(y_j - p_j) \tag{22}$$

and

$$\frac{\partial^2 E_c}{\partial y_j \partial y_k} = (g_j(y_j) - g_j'(y_j)(y_j - p_j))\delta_{k-j}. \tag{23}$$

The Hessian matrix is diagonal and, therefore, E_c is a concave function of $\mathbf{y}$ iff

$$g_j(\alpha)(p - \alpha) \tag{24}$$

is an increasing function of α for any value of scalar p between zero and one. ■

V. SSB Divergence Measures

A. SSB Divergence

Let us define

$$\begin{aligned} D(\mathbf{p}, \mathbf{y}) &= E_c(\mathbf{y}, \mathbf{p}) - E_c(\mathbf{p}, \mathbf{p}) \\ &= v(\mathbf{y}) - v(\mathbf{p}) + \sum_{i=1}^{L} \frac{\partial v}{\partial y_i}(p_i - y_i). \end{aligned} \tag{25}$$

Since $E_c(\mathbf{p}, \mathbf{p})$ does not depend on the network weights, minimizing the mean cost E_c is equivalent to minimizing D. Additionally, according to Kapur [15], D is a divergence measure between probability distributions $\mathbf{p}$ and $\mathbf{y}$ because it satisfies the following properties:

1) *Nonnegativity:* $D(\mathbf{p}, \mathbf{y}) \geq 0$;
2) *Identity:* $D(\mathbf{p}, \mathbf{y}) = 0$ iff $\mathbf{p} = \mathbf{y}$;
3) *Concavity:* $D(\mathbf{p}, \mathbf{y})$ is a concave function of $\mathbf{p}$.

Since $E_c(\mathbf{y}, \mathbf{p})$ is minimum at $\mathbf{y} = \mathbf{p}$, the verification of the nonnegativity and identity properties is easy. D is always concave as a function of $\mathbf{p}$, because v is convex and

$$\frac{\partial^2 D}{\partial p_i \partial p_j} = -\frac{\partial^2 v(\mathbf{p})}{\partial p_i \partial p_j}. \tag{26}$$

Moreover, as a direct consequence of (25), D is a concave function of $\mathbf{y}$ iff E_c is concave.

It is easy to show that the mathematical expressions for the divergence of the separable and symmetric SSB cost functions are given by

$$D(\mathbf{p}, \mathbf{y}) = \sum_{i=1}^{L} \int_{p_i}^{y_i} g_i(\alpha)(p_i - \alpha)\, d\alpha \tag{27}$$

and

$$D(\mathbf{p}, \mathbf{y}) = \sum_{i=1}^{L} \int_{p_i}^{y_i} g(\alpha)(p_i - \alpha)\, d\alpha, \tag{28}$$

respectively.

The last expression includes the Euclidean distance for $g(\alpha) = 2$ and the cross-entropy or the Kullback–Leibler divergence for $g(\alpha) = 1/\alpha$.

B. Mean Divergence and Insufficient Networks

Up to this point we have assumed that the classifier is able to compute the exact values of the *a posteriori* class probabilities for any input vector. Unfortunately, this is not a usual situation in practice and a question immediately arises: how do the SSB cost functions behave when they are applied to insufficient networks?

All the factors involved in the learning process influence the answer to this question: the architecture, the cost function, and the way the training samples are used during learning. Following Vapnik [7], if the learning process is *consistent*, the minimization of the accumulated cost

$$\overline{C} = \sum_{k=1}^{K} \{C(\mathbf{y}_k, \mathbf{d}_k)\} \tag{29}$$

is asymptotically equivalent (as K, the size of the training set, increases) to minimizing the mean cost

$$E\{C(\mathbf{y}, \mathbf{d})\} = \int E_c\{C(\mathbf{y}, \mathbf{p})|\mathbf{x}\} f_x(\mathbf{x})\, d\mathbf{x}. \tag{30}$$

The study of conditions under which the learning process is consistent is far beyond the scope of this paper. In any event, note that, according to (25), minimizing the mean cost is equivalent to minimizing the mean divergence $E\{D(\mathbf{y}, \mathbf{p})\}$, since $\mathbf{p}$ does not depend on the network parameters. Therefore, under the consistency assumption, the SSB cost function provides the *a posteriori* probability estimates which are, in the mean, the nearest to the true ones with respect to divergence measure D.

VI. Single Layer Perceptrons

In this section we will consider the application of SSB costs to estimate *a posteriori* probabilities using single layer perceptrons (SLP's). By SLP we mean any single-layer network with a nonlinear output activation function, satisfying the probability constraints used in this paper.

A. Single-Output Binary Classifiers

Consider the single-output SLP with soft decision

$$y = h(o) \tag{31}$$

where

$$o = \mathbf{w}^T \mathbf{x} \tag{32}$$

$\mathbf{w}$ is the weight matrix, and $h(.)$ is a monotonically increasing nonlinearity such that $0 \leq y \leq 1$. For a binary classification, condition (12) guarantees that the network output is an estimate of the *a posteriori* class probability. It is easy to see that the stochastic gradient learning rule for weights $\mathbf{w}$ is given by

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \rho g(y) h'(o)(d-y)\mathbf{x} \tag{33}$$

If h is a sigmoidal nonlinearity, $h'(o) = h(o)(1-h(o)) = y(1-y)$. This is a well known factor responsible for slow training when the quadratic cost function is used $(g(y) = 2)$, since the adaptation is slow when y is near zero or one (for example, see [19] or [25]). This is one of the justifications for using cross-entropy. In this case, $g(y) = y^{-1}(1-y)^{-1}$ (see [6]), $g(y)h'(o) = 1$, and

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \rho(d-y)\mathbf{x}. \tag{34}$$

Wittner and Denker [26] provide theoretical results showing that cross-entropy behaves better than the quadratic cost because it is a *well-formed* cost function. A cost-function is said to be well formed if it satisfies the following properties.

1) For all $y \in S$

$$s_d \frac{\partial C(y, d)}{\partial o} < 0$$

where

$$s_d = \begin{cases} 1 & d = 1 \\ -1 & d = 0 \end{cases} \tag{35}$$

(i.e., the learning rule does not push in the wrong direction).

2) There exists some $\epsilon > 0$ such that, if $s_d y < 0.5$

$$-s_d \frac{\partial C(y, d)}{\partial o} < \epsilon$$

(i.e., the learning rule keeps pushing if there is misclassification).

3) C is bounded below.

Since $\partial C(y, d)/\partial o = g(y)h'(o)$, which is the factor multiplying the error in (33), depends on both the cost function and the nonlinearity, what is in fact well formed is not the cost function, but the learning rule. An equivalent set of conditions for well-formed SSB leaning rules is given by the following proposition.

Proposition 1: An SSB learning rule for an SLP is well formed iff it satisfies the following properties.

1) $h'(o) > 0$ (i.e., the nonlinearity is monotonically increasing).
2) There exists some $\epsilon > 0$ such that $g(y)h'(o) > \epsilon$.

The proof is straightforward and we omit it here.

Note that if h is monotonically increasing, it is invertible and $h'(o) > 0$, and we can define an SSB cost function as

$$g(y) = \frac{1}{h'(h^{-1}(y))} \tag{36}$$

so that $g(y)h'(o) = 1$ and the learning rule (34) is satisfied. This proves the following proposition.

Proposition 2: If y is the output of an SLP with an increasing nonlinearity, learning rule (34) minimizes an SSB cost function.

Thus, irrespective of the exact form of the nonlinearity after the weighted sum of the input components, the previous learning rule can always be used to estimate probabilities (with arbitrary precision if the network can model the data profiles, or with minimum SSB divergence in a general case). This can be applied in analogical implementations of the SLP, where the exact expression of the activation function may be unknown [27].

B. Multioutput Single-Layer Networks

Amari [20] has shown that the particular advantages of using the cross entropy cost function to train perceptrons with a logistic function can be had in the multilevel case when the linear outputs o_i are fed to a softmax nonlinearity, i.e.,

$$y_i = \frac{\exp(o_i)}{\sum_{j=1}^{L} \exp(o_j)} \tag{37}$$

where $o_i = \mathbf{w}_i^T \mathbf{x}$ and the cross entropy cost function is used. It is easy to show that, in such a case, the weight updating rule is given by

$$\mathbf{w}_i(k+1) = \mathbf{w}_i(k) + \rho(d_i - y_i)\mathbf{x}. \tag{38}$$

The question about the SSB character of learning rule (38) arises immediately. This is answered with Theorem 5.

Theorem 5: Let $\mathbf{y}$ be the output of a SLP with nonlinearity $\mathbf{h}$. Learning rule

$$\mathbf{w}_i(k+1) = \mathbf{w}_i(k) + \rho(d_i - y_i)\mathbf{x} \tag{39}$$

minimizes an SSB cost function iff there exists a convex function $F(\mathbf{o})$ such that $\mathbf{y} = \mathbf{h}(\mathbf{o}) = \nabla_{\mathbf{o}} F$ (i.e., $\mathbf{h}$ is the gradient of a potential function) and its Hessian matrix $\mathbf{H}_F$ satisfies rank$(\mathbf{H}_F) = L - 1$.

Proof: See Appendix D. ■

The previous rules can only be applied to SLP's, which have a limited classification capability and, therefore, are not universal probability estimators. Learning rules for universal classifiers that are independent of the nonlinear activation function are being investigated by the authors.

VII. Discussion

A first question arises in connection with practical applications of the above theory. In particular, what differences exist between probability estimation and other Bayesian approaches? Minimizing an SSB cost for a learning machine is an alternative to the estimation of the *a posteriori* probabilities from a given model (or even a nonparametric formulation). Differences between these two approaches are obvious. If we have a model, we need only estimate its parameters, and this can be done by means of block algorithms in some cases. Training a machine usually requires gradient algorithms. On the other hand, the adequacy of the model is always an open question (which forces us to apply validation mechanisms that can lead to wrong results). A simple example serves to illustrate the situation.

Let us consider a binary Gaussian decision problem

$$\begin{aligned} f_i(\mathbf{x}) &= f(\mathbf{x}|d_i = 1) \\ &= \frac{1}{(2\pi\ \det(\mathbf{S}))^{N/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{u_i})^T\mathbf{S}^{-1}(\mathbf{x}-\mathbf{u_i})\right) \end{aligned} \tag{40}$$

where the classes are equiprobable. It is well known [2] that

$$P(d_i = 1|\mathbf{x}) = \frac{\exp(\mathbf{w}_i^T\mathbf{x} + b_i)}{\sum_{j=1}^{L} \exp(\mathbf{w}_j^T\mathbf{x} + b_j)} \tag{41}$$

where

$$\mathbf{w_i} = -\mathbf{S}^{-1}\mathbf{u_i} \tag{42}$$

$$b_i = -\tfrac{1}{2}\mathbf{u_i}^T\mathbf{S}^{-1}\mathbf{u_i} + \log(P(d_i = 1)). \tag{43}$$

Equation (41) shows that the SLP with a softmax activation function (or the logistic function in a binary case) is an SSB classifier of Gaussian clusters with the same covariance matrix.

Assume that the statistics of the class distributions are not known. The SLP can be trained using any learning rule minimizing an SSB cost; however, if we know that the distributions are Gaussian, density estimation is an alternative approach. Specifically, first the statistics of the data distributions [mean vectors $\mathbf{u_i}$, covariance matrix $\mathbf{S}$ and *a priori* probabilities $P(d_i = 1)$] are estimated, and then weight vector $\mathbf{w}$ is computed using (42) and (43).

However, there are many data distributions for which the single layer perceptron is an SSB classifier. For example, if the data are driven by conditional densities having forms

$$g(\mathbf{x})f_i(\mathbf{x}) \tag{44}$$

where $g(\mathbf{x})$ is some positive function independent of i, the optimal structure remains the same, but the estimation of probabilities under the hypothesis of Gaussian distributions will fail. Thus, the direct estimation approach is more general than that of density estimation. This can be shown with a simple one-dimensional example: assume that class distributions are driven by (44) with $\mu_1 = -\mu_2 = 3$, variance $\sigma = 1$ and $g(x) = a\ \sin^2(\pi x/\mu)$. Constant a is used to ensure a unit area density function. It is easy to show that, if $w = 2\mu/\sigma$,

$$P(1|\mathbf{x}) = \text{sigm}(wx). \tag{45}$$

Thus, a single output sigmoidal perceptron with a single weight is an SSB classifier for this problem. We have estimated network weight w following two different approaches.

1) Minimizing the cross entropy cost by means of learning rule (34).
2) Computing σ and μ and then using (42) and (43).

Fig. 2 represents the square weight error as learning proceeds, averaged over 300 realizations. The advantages of direct probability estimation over density estimation in this example are evident. The density estimation approach fails, because the actual distributions are not Gaussian; however, the learning rule minimizing the cross entropy finds the asymptotically optimal network parameter. This shows that SSB cost functions do not make unnecessary assumptions about the data distribution.

There is another important aspect from a practical point of view: how the "best" SSB cost for a particular problem can be determined? A general answer is not easy, and it is connected to some other aspects of training. In the following, we will restrict the discussion to the binary case, mainly for reasons of clarity.

We minimize

$$E\{C(y, d)\} = \int E\{C(y, d)|\mathbf{x}\} f_x(\mathbf{x})\, d\mathbf{x} \tag{46}$$

where $E\{C(y, d)|\mathbf{x}\}$ has the form shown in (12) for an SSB solution. It has been shown (in [4], for example) that the presence of $f_x(\mathbf{x})$ in (46) means that the minimization is emphasized for the regions at which the data density is higher. In the majority of practical situations (low error probability), this density is low around the classification surface, limiting the quality of the approach obtained for this border. By selecting an appropriate cost one can compensate for this difficulty in a way similar to the use of importance sampling to compute P_{FA} in many radar and communication problems. Specifically, $g(y)$ appears as a factor of $f_x(\mathbf{x})$ in $\partial E\{C(y, d)\}/\partial y$ (which must be zero to reach the solution, if the machine is general enough) and, consequently, selecting a high $g(y)$ around the border (when $y = 1/2$ in MAP classification) will help to improve the quality of the border and of the classification (at the expense of poorer estimates of the *a posteriori* probabilities in other regions, of course). Needless to say, this does not mean that the above selection will also yield fast convergence. Initially, the region of $\mathbf{x}$ to which $y = 1/2$ corresponds to depends on the initial values of the parameters (which are randomly selected in most procedures), and to apply $g(y)$ with its highest value around $y = 1/2$ would make the algorithm even slower. It seems much more reasonable to "accelerate" it by allowing $g(y)$ be higher where most the samples are, or at least constant (as for the quadratic cost). To increase the importance of error

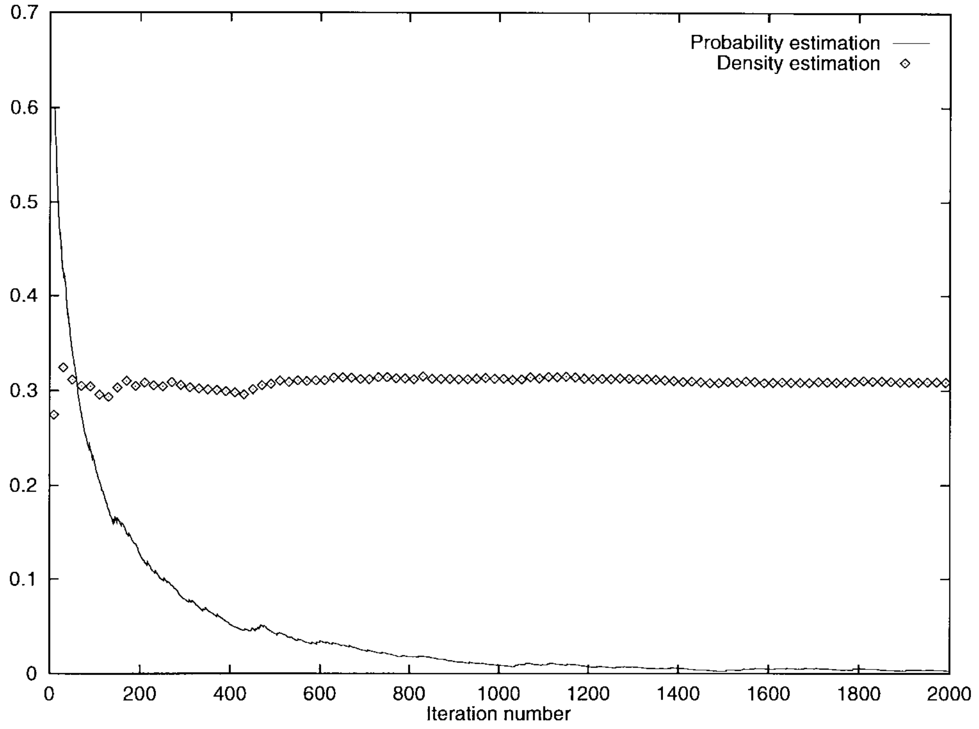

Fig. 2. Probability estimation versus density estimation. Evolution of the square weight error during learning. Dashed line: stochastic minimization of the cross entropy; diamonds: training via density estimation. Class distributions are driven by (44) with $\mu_1 = -\mu_2 = 3$, $\sigma = 1$, and $g(x) = a\ \sin^2(\pi x/\mu)$. Simulations show the average of 300 realizations.

by emphasizing factor $(y - d)$ (using $|y - d|^r$, $r > 1$, for example) also seems reasonable. This has to be changed for the last convergence steps if an SSB solution is desired. More analysis is needed to develop a solid theory of adapting the cost function to the convergence of the algorithm.

Another subject directly related to the above discussion, although it has not been analyzed from this point of view, is that of selecting samples. It was proposed first in [28], as a procedure according to which the samples that are difficult to learn are most frequently applied (after reaching a reasonable degree of convergence). Curiously, many subsequent selection strategies ([29] presents a significant number of options) take the perspective of selecting samples close to the decision borders. Of course, this is (qualitatively) equivalent to selecting an "equivalent" cost function, but is not as general as changing the sample distribution (as when applying importance sampling), or selecting samples in order to get a better definition of the decision borders, as implicitly done by quadratic (or linear) programming in minimizing combinations of performance and generalization measures to build support vector machines (see [7] and subsequent papers on the subject, such as [30]).

The use of a varying adaptation step can also be immediately related to the above discussion. A form $\eta(y, d)$ could be included as a factor in $\partial E\{C(y, d)|\mathbf{x}\}$, and a joint discussion will be possible. Most of the variable step rules (see [13], for a fairly complete presentation) could be analyzed from this point of view. However, we repeat that it is necessary to dedicate additional work to this line of research before establishing useful results and recommendations.

Cost selection also depends on other factors, including the network structure, the data distribution or the application requirements. If the classifier is not general enough, i.e., it cannot exactly map the class probabilities for any value of the network parameters, the optimal weight vector depends on the SSB cost. The requirements of the application can also influence the cost selection. For instance, when relative differences are more relevant than absolute differences, cross entropy could be preferred to quadratic cost, as the simulations reported in [19] show. A large comparative work has been carried out in the literature showing the advantages of the cross entropy over the quadratic cost. El-Jaroudi [19] notes that the reason for this may be the simplicity of the learning rule resulting when output logistic nonlinear activation functions are used [see (34), for an SLP], and Wittner and Denker [26] show that any gradient descent rule based on a well-formed learning rule converges to an optimal zero-error solution, provided that classes are separable. Moreover, they give examples of data distributions for which gradient descent based on the quadratic cost can converge to local minima. However, note that the cross entropy is advantageous over other cost functions in networks with sigmoid or softmax nonlinearities. Using other nonlinear activation functions, cross-entropy may not even yield well-formed learning rules. In any event, here we have demonstrated that learning rule (34) is universal, in the sense that it minimizes an SSB cost function independently of the nonlinear saturation used.

VIII. Conclusions

We have established necessary and sufficient conditions for SSB cost functions (i.e., those that provide estimates of the *a posteriori* probabilities of the classes) in networks whose outputs are consistent with probability laws. We have provided a general formula for SSB cost functions satisfying two important properties: separability and symmetry; showing that the quadratic cost and the cross entropy are separable, symmetric, and SSB.

We have paid special attention to the application of SSB cost functions to estimate probabilities using single layer perceptrons, providing an “iff” condition for an SSB cost to be well-formed for a single-output SLP. For both single and multioutput nets, we have presented a learning rule that always minimizes an SSB cost independent of the nonlinear activation function used.

The previous discussion shows that further work is necessary in order to determine the structure and behavior of the SSB cost family, and the relationship between *a posteriori* probability estimation and other Bayesian approaches to learning. Many applications could take advantage of an efficient *a posteriori* probability estimation algorithm.

Appendix A
Proof of Theorem 1

We can write the cost function as

$$C(\mathbf{y}, \mathbf{d}) = \sum_{i=1}^{L} d_i C_i(\mathbf{y}) \tag{47}$$

where

$$C_i(\mathbf{y}) = C(\mathbf{y}, \mathbf{d}_i). \tag{48}$$

Therefore

$$E_c = E\{C(\mathbf{y}, \mathbf{d})|\mathbf{x}\} = \sum_{i=1}^{L} p_i C_i(\mathbf{y}) \tag{49}$$

where

$$p_i = P(d_i = 1|\mathbf{x}), \qquad i = 1, \cdots, L \tag{50}$$

and $\mathbf{d_i}$ is the ith class unit vector with components $d_{i,j} = \delta_{i-j}$.

Because of (13), the network outputs are dependent variables. Using Lagrange multipliers, the minimum of $E_c(\mathbf{y}, \mathbf{p})$ as a function of $\mathbf{y}$ can be found minimizing

$$L(\mathbf{y}, \lambda) = E_c(\mathbf{y}, \mathbf{p}) + \lambda\left(\sum_{i=1}^{L} y_i - 1\right). \tag{51}$$

If C is SSB, there is a unique minimum at $\mathbf{y} = \mathbf{p}$; therefore, combining this with (49), we can write

$$\left.\frac{\partial L}{\partial y_j}\right|_{\mathbf{y}=\mathbf{p}} = \sum_{i=1}^{L} p_i \left.\frac{\partial C_i}{\partial y_j}\right|_{\mathbf{y}=\mathbf{p}} + \lambda = 0. \tag{52}$$

The set of differential equations given by (52) for every j should be satisfied by every vector $\mathbf{p} \in S$, so we can write

$$\sum_{i=1}^{L} y_i \frac{\partial C_i}{\partial y_j} + \lambda = 0. \tag{53}$$

These equations can be solved as follows: first, noting that

$$\begin{aligned} y_i \frac{\partial C_i}{\partial y_j} &= \frac{\partial}{\partial y_j}(y_i C_i) - \frac{\partial y_i}{\partial y_j} C_i \\ &= \frac{\partial}{\partial y_j}(y_i C_i) - \delta_{i-j} C_i. \end{aligned}$$

Equation (53) can be expressed as

$$\frac{\partial v}{\partial y_j} - C_j + \lambda = 0 \tag{54}$$

where

$$v = \sum_{i=1}^{L} y_i C_i. \tag{55}$$

Since (54) is valid for every j, we can write

$$\sum_{j=1}^{L} y_j\left(\frac{\partial v}{\partial y_j} - C_j + \lambda\right) = \sum_{j=1}^{L} y_j \frac{\partial v}{\partial y_j} - v + \lambda = 0 \tag{56}$$

$$\lambda = v - \sum_{j=1}^{L} y_j \frac{\partial v}{\partial y_j} \tag{57}$$

and, replacing λ in (54), we get

$$\begin{aligned} C_i &= \frac{\partial v}{\partial y_i} + v - \sum_{j=1}^{L} y_j \frac{\partial v}{\partial y_j} \\ &= v + \sum_{j=1}^{L} (\delta_{i-j} - y_j)\frac{\partial v}{\partial y_j}. \end{aligned} \tag{58}$$

Using (47) and (49)

$$C(\mathbf{y}, \mathbf{d}) = v + \sum_{j=1}^{L} (d_j - y_j)\frac{\partial v}{\partial y_j} \tag{59}$$

and

$$\begin{aligned} E_c(\mathbf{y}, \mathbf{p}) &= E\{C(\mathbf{y}, \mathbf{d})|\mathbf{x}\} \\ &= v + \sum_{j=1}^{L} (p_j - y_j)\frac{\partial v}{\partial y_j}. \end{aligned} \tag{60}$$

The convexity of v can be shown as follows:

$$\frac{\partial E_c}{\partial y_i} = \sum_{j=1}^{L} (p_j - y_j)\frac{\partial^2 v}{\partial y_i \, \partial y_j} \tag{61}$$

and

$$\frac{\partial^2 E_c}{\partial y_i \, \partial y_k} = \sum_{j=1}^{L} (p_j - y_j)\frac{\partial^3 v}{\partial y_i \, \partial y_j \, \partial y_k} - \frac{\partial^2 v}{\partial y_j \, \partial y_k}. \tag{62}$$

Since the stationary point is a minimum, $E_c(\mathbf{y}, \mathbf{p})$ must be a concave function of $\mathbf{y}$ at $\mathbf{y} = \mathbf{p}$; but, noting that

$$\left.\frac{\partial^2 E_c}{\partial y_i \, \partial y_k}\right|_{\mathbf{y}=\mathbf{p}} = -\frac{\partial^2 v}{\partial y_j \, \partial y_k} \tag{63}$$

we conclude that v is convex in S.

To show that $\mathbf{y} = \mathbf{p}$ is the unique stationary point, note that, at any stationary point

$$\frac{\partial E_c}{\partial y_i} = 0. \tag{64}$$

So, using (61), we have

$$\mathbf{H}_v(\mathbf{p}-\mathbf{y}) = \mathbf{0} \tag{65}$$

where $\mathbf{H}_v$ is the Hessian matrix of v. Since v is convex, $\mathbf{H}_v$ is nonsingular and, thus, the unique solution of (65) is $\mathbf{y}=\mathbf{p}$.

This proves that (15) gives a necessary condition. The proof of sufficiency is straightforward. If cost function is given by (59)–(61) are satisfied, so that $\mathbf{y}=\mathbf{p}$ is a stationary point of E_c; moreover, if v is convex in S

$$v(\mathbf{y}) - v(\mathbf{p}) > (\mathbf{y}-\mathbf{p})^T \nabla v \tag{66}$$

and thus

$$v(\mathbf{y}) - v(\mathbf{p}) - (\mathbf{y}-\mathbf{p})^T \nabla v(\mathbf{y}) = C(\mathbf{y}) - C(\mathbf{p}) > 0 \tag{67}$$

showing that $\mathbf{p}$ is a global minimum.

Appendix B
Proof of Theorem 2

If C is separable

$$\frac{\partial C}{\partial y_k} = c_k'(y_k) \tag{68}$$

and

$$\frac{\partial^2 C}{\partial y_k \, \partial y_l} = 0, \qquad k \neq l. \tag{69}$$

Noting that

$$\frac{\partial^2 C}{\partial y_k \, \partial y_l} = \sum_{i=1}^{L} (d_i - y_i) \frac{\partial^3 v}{\partial y_k \, \partial y_l \, \partial y_i} - \frac{\partial^2 v}{\partial y_k \, \partial y_l} \tag{70}$$

we have

$$\sum_{i=1}^{L} d_i \frac{\partial^3 v}{\partial y_k \, \partial y_l \, \partial y_i} = \sum_{i=1}^{L} y_i \frac{\partial^3 v}{\partial y_k \, \partial y_l \, \partial y_i} + \frac{\partial^2 v}{\partial y_k \, \partial y_l}, \qquad k \neq l. \tag{71}$$

The previous equality must hold for any target vector $\mathbf{d}$. For instance, taking $d_i = \delta_{i-m}$, $1 \le m \le L$, we get

$$\frac{\partial^3 v}{\partial y_k \, \partial y_l \, \partial y_m} = \sum_{i=1}^{L} y_i \frac{\partial^3 v}{\partial y_k \, \partial y_l \, \partial y_i} + \frac{\partial^2 v}{\partial y_k \, \partial y_l}. \tag{72}$$

Defining

$$f_{k,l}(\mathbf{y}) = \sum_{i=1}^{L} y_i \frac{\partial^3 v}{\partial y_k \, \partial y_l \, \partial y_m} + \frac{\partial^2 v}{\partial y_k \, \partial y_l} \quad 1 \le k, l \le L \tag{73}$$

we have

$$\frac{\partial^3 v}{\partial y_k \, \partial y_l \, \partial y_m} = f_{k,l}(\mathbf{y}). \tag{74}$$

Equation (74) holds for every m; thus, replacing it in (72), we arrive at

$$f_{k,l}(\mathbf{y}) = \left(\sum_{i=1}^{L} y_i\right) f_{k,l}(\mathbf{y}) + \frac{\partial^2 v}{\partial y_k \, \partial y_l}. \tag{75}$$

At any point $\mathbf{y} \in S$, (13) is verified and, therefore,

$$\frac{\partial^2 v}{\partial y_k \, \partial y_l} = 0 \qquad k \neq l \tag{76}$$

and

$$\frac{\partial C}{y_k} = (d_k - y_k) \frac{\partial^2 v}{\partial y_k^2}. \tag{77}$$

Using (68), we get

$$(d_k - y_k) \frac{\partial^2 v}{\partial y_k^2} = c_k'(y_k, d_k) \tag{78}$$

thus, at points $\mathbf{y} \in S$, $\partial^2 v / \partial y_k^2$ does not depend on y_l, $k \neq l$. Therefore, we can define

$$g_k(y_k) = -\frac{\partial^2 v}{\partial y_k^2} \tag{79}$$

and, consequently

$$c_k(y_k, d_k) = \int_{r_k(\mathbf{d})}^{y_k} (\alpha - d_k) g_k(\alpha)\, d\alpha + s_k(\mathbf{d}) \tag{80}$$

where r_k and s_k are arbitrary functions of $\mathbf{d}$ which cannot depend on $\mathbf{y}$. For convenience, we can express

$$c_k(y_k, d_k) = \int_{d_k}^{y_k} (\alpha - d_k) g_k(\alpha)\, d\alpha + \int_{r_k(\mathbf{d})}^{d_k} (\alpha - d_k) g_k(\alpha)\, d\alpha + s_k(\mathbf{d}) \tag{81}$$

therefore

$$C(\mathbf{y}, \mathbf{d}) = \sum_{k=1}^{L} \int_{d_k}^{y_k} (\alpha - d_k) g_k(\alpha)\, d\alpha + r(\mathbf{d}) \tag{82}$$

where

$$r(\mathbf{d}) = \sum_{k=1}^{L} \int_{r_k(\mathbf{d})}^{d_k} (\alpha - d_k) g_k(\alpha)\, d\alpha + \sum_{k=1}^{L} s_k(\mathbf{d}). \tag{83}$$

Since there are no restrictions imposed to r_k and s_k, r is an arbitrary function of $\mathbf{d}$, and this completes the proof of Theorem 2.

Appendix C
Proof of Theorem 3

Consider permutation function $\mathbf{Q}$ that only changes components m and n in $\mathbf{y}$, $m \neq n$

$$Q_i(\mathbf{y}) = \begin{cases} y_m, & \text{if } i = n \\ y_n, & \text{if } i = m \\ y_i, & \text{otherwise.} \end{cases} \tag{84}$$

Since C is symmetric and separable, (17) and (18) hold. Therefore

$$\sum_{i=1}^{L} \int_{d_i}^{y_i} g_i(\alpha)(\alpha - d_i)\, d\alpha + r(\mathbf{d}) = \sum_{i=1}^{L} \int_{Q_i(\mathbf{d})}^{Q_i(\mathbf{y})} g_i(\alpha)(\alpha - Q_i(\mathbf{d}))\, d\alpha + r(\mathbf{Q}(\mathbf{d})). \tag{85}$$

Using (84) we arrive at

$$\begin{aligned}&\int_{d_m}^{y_m} g_m(\alpha)(\alpha-d_m)\,d\alpha+\int_{d_n}^{y_n} g_n(\alpha)(\alpha-d_n)\,d\alpha+r(\mathbf{d})\\&\quad=\int_{d_m}^{y_m} g_n(\alpha)(\alpha-d_m)\,d\alpha+\int_{d_n}^{y_n} g_m(\alpha)(\alpha-d_n)\,d\alpha\\&\qquad+r(\mathbf{Q}(\mathbf{d})).\end{aligned}\tag{86}$$

The previous equality must hold for any probability vectors $\mathbf{y}$ and $\mathbf{d}$. Now, we have to consider the cases $L>2$ and $L=2$ separately. In the former case, we can choose $\mathbf{y}$ and $\mathbf{d}$ such that $d_m=d_n=y_n=0$; then $\mathbf{Q}(\mathbf{d})=\mathbf{d}$ and

$$\int_0^{y_m}(g_m(\alpha)-g_n(\alpha))\alpha\,d\alpha=0\tag{87}$$

for every y_m. Thus, $g_m(\alpha)=g_n(\alpha)$ for every n, and defining $g(\alpha)=g_n(\alpha)$, we arrive at (19).

This proves the theorem for $L>2$. Consider the case $L=2$ (the previous proof is not valid because it is not possible to take $d_m=d_n=0$ unless $m=n$). Let us take $m=1$, $n=2$ and $y=y_1=1-y_2$ if C is separable

$$C((y,1-y),(d,1-d))=C((1-y,y),(1-d,d)).\tag{88}$$

Now, taking the first derivative with respect to y,

$$\begin{aligned}&g_1(y)(y-d)-g_2(1-y)((1-y)-(1-d))\\&\quad=-g_1(1-y)((1-y)-(1-d))+g_2(y)(y-d)\end{aligned}\tag{89}$$

and, for $d=0$

$$g_1(y)+g_2(1-y)=g_1(1-y)-g_2(y).\tag{90}$$

Let us define

$$g(y)=\tfrac{1}{2}(g_1(y)+g_2(1-y)).\tag{91}$$

Using the fact that $g(y)=g(1-y)$, we can develop the binary cost as follows:

$$\begin{aligned}&C((y,1-y),(d,1-d))\\&\quad=\int_d^y g_1(\alpha)(\alpha-d)\,d\alpha+\int_{1-d}^{1-y} g_2(\alpha)(\alpha-(1-d))\,d\alpha\\&\qquad+r(\mathbf{d})\\&\quad=\int_d^y g_1(\alpha)(\alpha-d)\,d\alpha+\int_d^y g_2(1-\alpha)(\alpha-d)\,d\alpha\\&\qquad+r(\mathbf{d})\\&\quad=\int_d^y 2g(\alpha)(\alpha-d)\,d\alpha+r(\mathbf{d})\\&\quad=\int_d^y g(\alpha)(\alpha-d)\,d\alpha+\int_d^y g(1-\alpha)(\alpha-d)\,d\alpha\\&\qquad+r(\mathbf{d})\\&\quad=\int_d^y g(\alpha)(\alpha-d)\,d\alpha+\int_{1-d}^{1-y} g(\alpha)(\alpha-(1-d))\,d\alpha\\&\qquad+r(\mathbf{d}).\end{aligned}\tag{92}$$

The final expression is equivalent to (19) for $L=2$.

Appendix D
Proof of Theorem 5

The stochastic gradient learning rule for a single-layer multioutput network is

$$\begin{aligned}\mathbf{w}_i(k+1)&=\mathbf{w}_i(k)-\rho\nabla_{\mathbf{w}_i}C &(93)\\&=\mathbf{w}_i(k)-\rho\sum_{k=1}^{L}\frac{\partial C}{\partial o_k}\nabla_{\mathbf{w}_i}o_k &(94)\\&=\mathbf{w}_i(k)-\rho\frac{\partial C}{\partial o_i}\mathbf{x}. &(95)\end{aligned}$$

Let us assume that (38) minimizes C and that C is SSB. Using (38) and (95) we get

$$\frac{\partial C}{\partial o_i}=y_i-d_i.\tag{96}$$

Defining $C_i(\mathbf{y})=C(\mathbf{y},\mathbf{d_i})$ as in (48), we find

$$C(\mathbf{y},\mathbf{d})=\sum_{k=1}^{L}d_kC_k(\mathbf{y})\tag{97}$$

and

$$\sum_{k=1}^{L}d_k\frac{\partial C_k}{\partial o_i}=y_i-d_i.\tag{98}$$

The last equality must hold for every target vector $\mathbf{d}$. For instance, if $d_i=\delta_{i-j}$ we get

$$\frac{\partial C_j}{\partial o_i}=y_i-\delta_{i-j}.\tag{99}$$

Note that

$$\frac{\partial^2 C_j}{\partial o_i\,\partial o_m}=\frac{\partial y_i}{\partial o_m}.\tag{100}$$

Since second derivatives are independent of the derivation order, we find

$$\frac{\partial y_i}{\partial o_m}=\frac{\partial y_m}{\partial o_i}.\tag{101}$$

Thus, $\mathbf{y}=\mathbf{h}(\mathbf{o})$ is the gradient of a potential function

$$\mathbf{y}=\nabla_{\mathbf{o}}F.\tag{102}$$

Moreover, F is convex; this can be shown as follows. First, note that

$$\frac{\partial C}{\partial o_k}=\sum_{j=1}^{L}\frac{\partial C}{\partial y_j}\frac{\partial y_j}{\partial o_k}\tag{103}$$

so that, since

$$\frac{\partial C}{\partial y_i}=\sum_{i=1}^{L}(d_i-y_i)\frac{\partial^2 v}{\partial y_i\,\partial y_j}\tag{104}$$

and, using (96), we arrive at

$$y_k-d_k=\sum_{j=1}^{L}\left(\sum_{i=1}^{L}(d_i-y_i)\frac{\partial^2 v}{\partial y_i\,\partial y_j}\right)\frac{\partial y_j}{\partial o_k}.\tag{105}$$

Taking into account

$$\frac{\partial y_j}{\partial o_k} = \frac{\partial^2 F}{\partial o_j \, \partial o_k} \tag{106}$$

we can express the previous equation in matrix form as

$$(\mathbf{d} - \mathbf{y})^T (\mathbf{H_v H_F} + \mathbf{I}) = \mathbf{0} \tag{107}$$

where $\mathbf{H_v}$ and $\mathbf{H_F}$ are the Hessian matrices of $v(\mathbf{y})$ and $F(\mathbf{o})$, respectively. As this must be true for any target vector $\mathbf{d}$, we can write

$$\sum_{i=1}^{L} a_i (\mathbf{d}_i - \mathbf{y})^T (\mathbf{H_v H_F} + \mathbf{I}) = \mathbf{0} \tag{108}$$

for any real numbers a_i, $i = 1, \cdots, L$. If we take a_i such as $\sum_{i=1}^{L} a_i = 0$, it is easy to see that the previous equality can be written as

$$\mathbf{a}^T (\mathbf{H_v H_F} + \mathbf{I}) = \mathbf{0} \tag{109}$$

where $\mathbf{a} = (a_1, \cdots, a_L)^T$ is any vector in the subspace $A = \{\mathbf{a} | \sum_{i=1}^{L} a_i = 0\}$. This has two consequences.

1) Note that

$$\begin{aligned}\sum_{i=1}^{L} \frac{\partial^2 F}{\partial o_m \, \partial o_i} &= \sum_{i=1}^{L} \frac{\partial y_i}{\partial o_i} \\ &= \frac{\partial}{\partial o_i} \sum_{i=1}^{L} y_i = 0 \end{aligned} \tag{110}$$

therefore, the column vectors of matrix $\mathbf{H_F}$ are in A, and

$$\mathbf{H_F H_v H_F} + \mathbf{H_F} = 0. \tag{111}$$

Since v is convex in S, $\mathbf{H_v}$ is definite positive and, thus, for every vector $\mathbf{u}$

$$\mathbf{u}^T \mathbf{H_F H_v H_F u} \geq 0 \tag{112}$$

and thus

$$\mathbf{u}^T \mathbf{H_F u} \leq 0 \tag{113}$$

showing that $\mathbf{H_F}$ is semidefinite negative and F is concave (although no strictly concave).

2) Let us define matrix $\mathbf{B} = \mathbf{H_v H_F} + \mathbf{I}$; thus, $\mathbf{a}^T \mathbf{B} = 0$ for any vector $\mathbf{a} \in A$. As $\dim(A) = L - 1$, we have that rank$(\mathbf{B}) = 1$. Moreover, since

$$\mathbf{I} = \mathbf{H_v H_F} - \mathbf{B} \tag{114}$$

and rank$(I) = L$, we get

$$L \leq \text{rank}(\mathbf{H_v H_F}) + \text{rank}(\mathbf{B}) = \text{rank}(\mathbf{H_v H_F}) + 1. \tag{115}$$

Since $\mathbf{H_v}$ is invertible, rank$(\mathbf{H_F}) \geq L - 1$. But, as the columns in $\mathbf{H_F}$ are all in A, rank$(\mathbf{H_F}) < L$, and we conclude that

$$\text{rank}(\mathbf{H_F}) = L - 1. \tag{116}$$

We now prove the converse of the theorem. Let us assume that $\mathbf{y} = \mathbf{h}(\mathbf{o})$ is the gradient of a potential function, $F(\mathbf{o})$, and rank$(\mathbf{H_F}) = L - 1$. Let us define

$$c_j(\mathbf{o}) = o_j - F(\mathbf{o}). \tag{117}$$

In the following, we demonstrate that

$$C(\mathbf{o}, \mathbf{d}) = \sum_{j=1}^{L} d_j c_j(\mathbf{o}) \tag{118}$$

is an SSB cost leading to rule (38). Since

$$\frac{\partial C(\mathbf{o}, \mathbf{d})}{\partial o_i} = \sum_{j=1}^{L} d_j \frac{\partial c_j}{\partial o_i} = \sum_{j=1}^{L} d_j (\delta_{j-i} - y_i) = d_i - y_i \tag{119}$$

learning rule (38) minimizes cost $C(\mathbf{o}, \mathbf{d})$ with a minimum when

$$\frac{\partial E\{C(\mathbf{o}, \mathbf{d}) | \mathbf{x}\}}{\partial o_i} = E\left\{ \frac{\partial C(\mathbf{o}, \mathbf{d})}{\partial o_i} \middle| \mathbf{x} \right\} = p_i - y_i = 0 \tag{120}$$

i.e., when outputs are probabilities. Note, however, that we have defined the SSB costs as functions of $\mathbf{y}$. Next, we show that C is, in fact, a function of $\mathbf{y}$. First, note that

$$\sum_{i=1}^{L} y_i = \sum_{i=1}^{L} \frac{\partial F}{\partial o_i} = (\nabla_{\mathbf{o}} F)^T \mathbf{u} = 1 \tag{121}$$

where $\mathbf{u} = (1, 1, \cdots, 1)/\sqrt{L}$ is a unit vector. Thus, the directional derivatives of F along the lines driven by $\mathbf{u}$ are constant, so that

$$F(\mathbf{o} + \lambda \mathbf{u}) - F(\mathbf{o}) = \frac{\lambda}{\sqrt{L}} \tag{122}$$

for every λ. According to this and, using (117)

$$\begin{aligned} c_j(\mathbf{o} + \lambda \mathbf{u}) - c_j(\mathbf{o}) &= \lambda u_i - (F(\mathbf{o} + \lambda \mathbf{u}) - F(\mathbf{o})) \\ &= \frac{\lambda}{\sqrt{L}} - \frac{\lambda}{\sqrt{L}} \\ &= 0. \end{aligned} \tag{123}$$

As rank$(\mathbf{H_F}) = L - 1$ at every point, F is strictly convex in the vector subspace A of all vectors that are orthogonal to $\mathbf{u}$. Therefore, there cannot be any vector $\mathbf{a} \notin \{\lambda \mathbf{u}, \lambda \in R\}$ such that $h(\mathbf{o}) = h(\mathbf{o} + \mathbf{a})$. Thus, for every $\mathbf{o_a}$ and $\mathbf{o_b}$ such that $h(\mathbf{o_a}) = h(\mathbf{o_b}) = \mathbf{y}$, we find $C(\mathbf{o_a}, \mathbf{d}) = C(\mathbf{o_b}, \mathbf{d})$, which demonstrates that C can be expressed as a unique function of $\mathbf{y}$ and $\mathbf{d}$.

REFERENCES

[1] H. L. Van Trees, *Detection, Estimation and Modulation Theory,* vol. I New York: Wiley, 1968.

[2] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* New York: Wiley, 1973.

[3] B. Pearlmutter and J. Hampshire, "Equivalence proofs for multilayer perceptron classifiers and the Bayesian discriminant function," in *Proc. 1990 Connectionist Models Summer School.* San Diego, CA: Morgan Kauffmann, 1990.

[4] D. W. Ruck, S. K. Rogers, M. Kabrisky, M. E. Oxley, and B. W. Suter, "The multilayer perceptron as an approximation to a Bayes optimal discriminant function," *IEEE Trans. Neural Networks,* vol. 1, pp. 296–298, Dec. 1990.

[5] M. D. Richard and R. P. Lippmann, “Neural network classifiers estimate Bayesian *a posteriori* probabilities,” *Neural Comput.,* vol. 3, no. 4, pp. 461–483, 1991.
[6] J. W. Miller, R. Goodman, and P. Smyth, “Objective functions for probability estimation,” in *Proc. Int. Joint Conf. Neural Networks,* 1991, vol. I, pp. 881–886.
[7] V. N. Vapnik, *The Nature of the Statistical Learning Theory.* New York: Springer Verlag, 1995.
[8] S. I. Gallant, *Neural Network Learning and Expert Systems.* Cambridge, MA: MIT Press, 1993.
[9] S. Y. Kung and J. S. Taur, “Decision-based neural networks with signal/image classification applications,” *IEEE Trans. Neural Networks,* vol. 6, pp. 170–181, Jan. 1995.
[10] V. P. Roychowdhury, K.-Y. Siu, and T. Kailath, “Classification of linearly nonseparable patterns by linear threshold elements,” *IEEE Trans. Neural Networks,* vol. 6, pp. 318–331, Mar. 1995.
[11] B. A. Telfer and H. H. Szu, “Implementing the minimum-missclassification-error energy function for target recognition,” in *Proc. 1992 Int. Conf. Neural Networks,* Baltimore, MD, vol. IV, 1992, pp. 214–219.
[12] ——, “Energy functions for minimizing missclassification error with minimum-complexity networks,” *Neural Networks,* no. 7, pp. 809–818, 1994.
[13] A. Cichocki and R. Unbenhauen, *Neural Networks for Optimization and Control.* Baffins Lane, U.K.: Wiley, 1993.
[14] P. S. Neelakanta, S. Abusalah, D. de Groff, R. Sudhakar, and J. C. Park, “Csiszár generalized error measures for gradient-descent-based optimizations in neural networks using the backpropagation algorithm,” *Connection Sci.,* vol. 8, no. 1, pp. 79–114, 1996.
[15] J. N. Kapur and H. K. Kesavan, *Entropy Optimization Principles with Applications.* San Diego, CA: Academic, 1993.
[16] J. J. Hopfield, “Learning algorithms and probability distributions in feed-forward and feed-back networks,” *Proc. Nat. Academy Sci. USA,* vol. 84, pp. 8429–8433, 1987.
[17] G. E. Hinton, “Connectionist learning procedures,” *Artificial Intell.,* vol. 40, pp. 185–234, 1989.
[18] J. Cid-Sueiro and A. R. Figueiras-Vidal, “Digital equalization using modular neural networks: An overview,” in *Proc. 7th. Int. Thyrrhenian Workshop Digital Commun.,* Viareggio, Italy, Sept. 1995, pp. 337–345.
[19] A. El-Jaroudi and J. Makhoul, “A new error criterion for posterior probability estimation with neural nets,” in *Proc. Int. Joint Conf. Neural Networks,* San Diego, CA, 1990, vol. III, pp. 185–192.
[20] S. Amari, “Backpropagation and stochastic gradient descent method,” *Neurocomput.,* no. 5, pp. 185–196, June 1993.
[21] E. A. Wan, “Neural network classification: A Bayesian interpretation,” *IEEE Trans. Neural Networks,* vol. 1, pp. 303–305, Dec. 1990.
[22] J. Billa and A. El-Jaroudi, “A method of generating objective functions for probability estimation,” *Eng. Applicat. Artificial Intell.,* vol. 9, no. 2, pp. 203–208, Apr. 1996.
[23] I. M. Elfadel and J. L. Wyatt, Jr., “The ‘softmax’ nonlinearity: Derivation using statistical mechanics and useful properties as a multiterminal analog circuit element,” in *Advances in Neural Information Processing Systems,* J. D. Cowan, G. Tesauro, and J. Alspector, Eds. San Mateo, CA: Morgan Kaufmann, 1994, vol. 6, pp. 882–887.
[24] S. Haykin, *Adaptive Filter Theory,* 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.
[25] T. Adali, X. Liu, and M. K. Sönmez, “Conditional distribution learning with neural networks and its application to channel equalization,” *IEEE Trans. Signal Processing,* vol. 45, pp. 1051–1064, 1997.
[26] B. S. Wittner and J. S. Denker, “Strategies for teaching layered neural networks classification tasks,” in *Neural Inform. Processing Syst.,* W. V. Oz and M. Yannakakis, Eds., Denver, CO, 1988, pp. 850–859.
[27] M. I. Elmasry, Ed., *VLSI Artificial Neural Networks Engineering.* Norwell, MA: Kluwer, 1994.
[28] P. W. Munro, “Repeat until bored: A pattern selection strategy,” in *Advances in Neural Information Processing Systems,* J. E. Moody, S. J. Hanson, and R. P. Lippmann, Eds., vol. 4. San Mateo, CA: Morgan Kaufmann, 1992, pp. 1001–1008.
[29] C. Cachin, “Pedagogical pattern selection strategies,” *Neural Networks,* vol. 7, no. 1, pp. 175–181, 1994.
[30] B. Schölkopf *et al.,* “Comparing support vector machines with Gaussian kernels to radial basis function classifiers,” *IEEE Trans. Signal Processing,* vol. 45, pp. 2758–2765, 1997.

Jesús Cid-Sueiro (S’92–M’95) received the bachelor’s degree in telecommunication engineering from the University of Vigo, Spain, in 1990, and the Ph.D. degree from the Technical University of Madrid, Spain, in 1994.

Since 1996, he has been an Associate Professor at the Department of Signal Theory, Communications and Computer Science, University of Valladolid, Spain. His main research interests include statistical learning theory, neural networks and their applications to communications, image processing, and education.

Juan Ignacio Arribas was born in Valladolid, Spain, in 1973. He received the B.S. and M.S. degrees, both in electrical engineerin from the University of Valladolid, Valladolid, Spain, in 1994 and 1996, respectively.

In 1996, he joined the Department of Signal Theory, Communications and Computer Science at the College of Engineering, University of Valladolid, Valladolid, Spain, where he has been working as a Research Associate. His current research topics include statistical signal processing, estimation theory, image processing, medical imaging, and their applications to medical diagnoses and communications.

Sebastián Urbán-Muñoz received the telecommunication engineering degree from the University of Seville, Spain, in 1996, and is now pursuing the Ph.D. degree from the University Carlos III of Madrid.

Since 1997, he has been an Asset-Management Quantitative Analyst at Banco Bilbao Vizcaya Group. His main research interests include neural networks and their applications to financial engineering, computational finance, and data mining.

Aníbal R. Figueiras-Vidal (S’74–M’76–SM’84) received the Telecomm Engineer degree from Universidad Politécnica de Madrid Spain, in 1973, and the Doctor degree in 1976 from Universidad Politécnica de Barcelona, Spain.

He is a Professor in Signal Theory and Communications at Universidad Carlos III de Madrid. His research interests are digital signal processing, digital communications, neural networks, and learning theory.