A Universal Learning Rule That Minimizes Well-Formed Cost Functions

Inma Mora-Jiménez, *Member, IEEE,* and Jesús Cid-Sueiro, *Member, IEEE*

***Abstract*—In this paper, we analyze stochastic gradient learning rules for posterior probability estimation using networks with a single layer of weights and a general nonlinear activation function. We provide necessary and sufficient conditions on the learning rules and the activation function to obtain probability estimates. Also, we extend the concept of well-formed cost function, proposed by Wittner and Denker, to multiclass problems, and we provide theoretical results showing the advantages of this kind of objective functions.**

***Index Terms*—Generalized soft perceptron (GSP), stochastic learning rule, strict sense Bayesian (SSB) cost function, well-formed cost function.**

I. Introduction

DURING the last years, a wide number of researchers have investigated the factors that influence in the probabilistic interpretation of the outputs of artificial neural networks (ANNs) [1], [3], [4], [14], [16]. When dealing with recognition tasks, it is known that, from a statistical point of view, the ANN soft outputs can be associated with a reliability measure of the classification process. This extra-information could be appropriated in cases that are not "clear" enough, being specially relevant in problems such as medical diagnosis or financial applications.

It is known that if the cost function to optimize is strongly multimodal (what is frequent in real world problems), stochastic learning rules are more suitable than batch-processing methods, since they prevent the solution to get stuck in local minima. This is the reason why stochastic gradient learning rules have been widely applied to solve optimization problems in many fields.

Algorithms based on stochastic gradient minimization present nice convergence properties and usually provide simple learning rules, not to mention the reduction on computational burden (compared with batch learning schemes). The form and behavior of the learning rule depends on two main factors: the selection of the cost function and the network structure. There is a strong relationship between these components. As an example, the advantages of the cross entropy over the square error as an objective function for optimization has been analyzed by several authors [1], [21], but the theoretical analysis carried out by Wittner and Denker [21] suggests that at least part of the advantages come from the particular form of the learning rule when the cross entropy is used to train networks with logistic activation functions.

Manuscript received March 20, 2003; revised December 20, 2004. This work was supported in part by CICYT under Grant TIC2002-03713 and in part by CAM under Grants GR/SAL/0471/2004 and 07T/0017/2003-1.

The authors are with the Department of Signal Theory and Communications, University Carlos III de Madrid, 28911 Leganés-Madrid, Spain (e-mail: inmoji@tsc.uc3m.es).

Digital Object Identifier 10.1109/TNN.2005.849839

Consider a Single Layer Perceptron with a logistic activation function whose output is given by $y = (1 + \exp(-\mathbf{w}^T\mathbf{x}))^{-1}$, where $\mathbf{x}$ is the sample vector and $\mathbf{w}^T$ represents the transposed vector of $\mathbf{w}$ (network parameters). It is easy to show that the stochastic gradient minimization of the cross entropy [4] for this network is given by learning rule

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

where $d \in \{0, 1\}$ is class label of input pattern $\mathbf{x}$, ρ is the learning step, and k is the iteration number. Note that (1) is perhaps the simplest learning rule that can be used for training a classifier in a supervised way (as in the perceptron rule [3], [12], the correction term is proportional to both the error and the input value). Despite (or thanks to) its simplicity, rule (1) has many advantages over other rules:

First, it can be shown that (1) guarantees that the parameter vector $\mathbf{w}$ obtained in this way belongs to a zero error solution in linear separable problems. Moreover, the analysis in [21] suggests that this learning rule finds zero error solutions for a wide range of single layer networks with different activation functions.

Second, in [4] it is proved that (1) is a *universal* learning rule for posterior probability estimation, in the sense that, for any strictly increasing activation function, the network output provides estimates of posterior probabilities ($y = P\{d = 1 \mid \mathbf{x}\}$).

In summary, there exists a strong relationship among the cost function and the activation function that is essential to the well-formed behavior of the learning rule. Exploring these connections is the main purpose of our work.

This paper analyzes theoretically the extension of properties mentioned previously to multiclass schemes and multioutput neural networks with arbitrary approximation capability. To do so, we analyze a learning rule that is a direct extension of (1) to multiclass problems. We show that under some conditions on the activation function, the learning rule provides estimates of the posterior class probabilities.

We also extend the definition of well-formed cost function by Wittner and Denker [21] to multiclass problems and show that a well-formed cost has also important advantages for multiclass problems. Finally, we show a multiclass extension of learning rule (1), that is also well-formed.

The paper is organized as follows. Section II describes the scenario. In Section III, we review the problem of estimating posterior probabilities with ANNs. Universal learning rules for probability estimation are discussed in Section IV and Section V proposes a multiclass extension of the concept of well-formed cost functions. An illustrative experiment is carried out in Section VI. Finally, we state some conclusions and suggest further work.

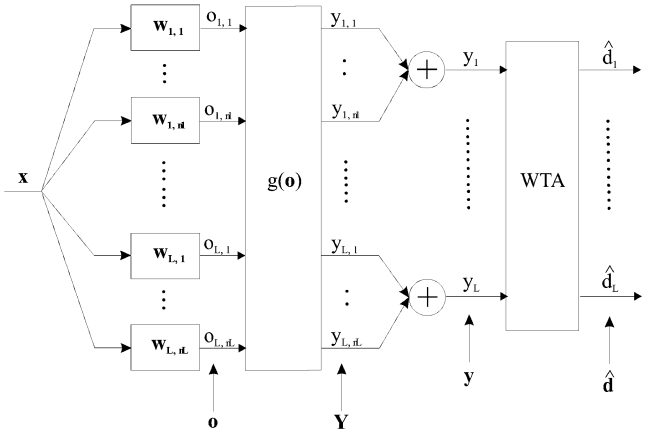

Fig. 1. Scheme of a GSP network with L classes and n_l outputs per class.

II. Problem Statement

A. Generalized Soft Perceptron

Fig. 1 shows the structure of a generalized soft perceptron (GSP) for an L-class problem. It consists of a linear layer with parameters $\mathbf{w}_{ij}$ $(i = 1, \ldots, L; j = 1, \ldots, n_i)$, where j indexes the number of "subclasses" or filters for the ith class, a nonlinear activation $\mathbf{g}(\cdot)$ function and a (unweighted) sum layer. For any input $\mathbf{x} \in \mathbb{R}^D$, the GSP computes one soft decision per class, given by

$$y_i = \sum_{j=1}^{n_i} y_{ij} \tag{2}$$

where y_{ij} are the outputs of nonlinear activation function $\mathbf{g}$

$$y_{ij} = g_{ij}(\mathbf{W}^T\mathbf{x}) \tag{3}$$

where parameter matrix

$$\mathbf{W} = (\mathbf{w}_{1,1}, \ldots \mathbf{w}_{1,n_1}, \ldots, \mathbf{w}_{L,1}, \ldots, \mathbf{w}_{L,n_L}) \tag{4}$$

encompasses all parameter vectors and g_{ij} represents the ijth component of function $\mathbf{g}$.

In the following, we will express the network equations in vector form: defining vectors $\mathbf{y}, \mathbf{Y}$, and $\mathbf{o}$ with components $\{y_i\}, \{y_{ij}\}$, and $\{o_{ij}\}$, respectively, the network equations can be written as

$$\mathbf{y} = \mathbf{U}\mathbf{Y} \tag{5}$$

where

$$U = \begin{pmatrix} \overbrace{1 \cdots 1}^{n_1} & \overbrace{0 \cdots 0}^{n_2} & \cdots & \overbrace{0 \cdots 0}^{n_L} \\ 0 \cdots 0 & 1 \cdots 1 & \cdots & 0 \cdots 0 \\ \cdots\cdots & \cdots\cdots & \cdots & \cdots\cdots \\ 0 \cdots 0 & 0 \cdots 0 & \cdots & 1 \cdots 1 \end{pmatrix} \tag{6}$$

$$\mathbf{Y} = \mathbf{g}(\mathbf{o}) \tag{7}$$

and

$$\mathbf{o} = \mathbf{W}^T\mathbf{x}. \tag{8}$$

A winner-take-all (WTA) network is used for *hard* decision, assigning a "1" to the class with the highest input y_i and zero to the rest. Therefore, components of output decision vector $\hat{\mathbf{d}}$ are given by $\hat{d}_i = \delta_{j-i}$, where δ is the Kronecker delta and $j = \arg\max_i\{y_i, i = 1, \ldots, L\}$. We consider there is an error if $\hat{\mathbf{d}}$ is different from target vector $\mathbf{d}$ with components δ_{c-i}, where c is the class the observation belongs to.

B. Probabilistic Activation Functions

Since we are interested in estimating posterior probabilities, we should impose several constraints on the activation function. In general, we will say that, $\mathbf{Y} = \mathbf{g}(\mathbf{o})$ is a *probabilistic* activation function if it satisfies two main properties.

Property P1: For any input vector $\mathbf{o}$, activation outputs y_{ij} are nonnegative

$$0 \le y_{ij} \le 1. \tag{9}$$

Property P2: For any input vector $\mathbf{o}$, activation outputs y_{ij} sum up to 1

$$\sum_{i=1}^{L}\sum_{j=1}^{n_i} y_{ij} = 1. \tag{10}$$

Thus, a probabilistic function maps every element of $\mathbb{R}^N$ into probability space $\mathcal{P}_N = \{\mathbf{P} \in \mathbb{R}^N : 0 \le p_{ij} \le 1, \sum_{i=1}^{L}\sum_{j=1}^{n_i} p_{ij} = 1\}$, where $N = \sum_{i=1}^{L} n_i$. For this reason, we can interpret outputs y_{ij} and y_i as probabilities.

There are two other properties that may be of our interest.

Property P3: $\mathbf{g}$ does not change the size ranking, i.e., for any i, j, k, l

$$o_{ij} > o_{kl} \Leftrightarrow y_{ij} > y_{kl}. \tag{11}$$

Property P4: $\mathbf{g}$ maps $\mathbb{R}^N$ over the complete probability space $\mathcal{P}_N$ in such a way that

$$o_{ij} \to \infty \Rightarrow y_{ij} \to 1 \tag{12}$$

$$o_{ij} \to -\infty \Rightarrow y_{ij} \to 0. \tag{13}$$

C. Softmax Activation Function

A particular GSP network is obtained when $\mathbf{g}$ is the softmax function with components y_{ij} given by

$$y_{ij} = \frac{\exp(o_{ij})}{\sum_{s=1}^{L}\sum_{t=1}^{n_s}\exp(o_{st})}. \tag{14}$$

It is immediate to see that the softmax satisfies properties P1 to P4 mentioned previously. The GSP with softmax activation function can approximate any posterior probability map. This can be shown as follows: consider we can model the samples assigned to each filter ij as a Gaussian distribution with mean $\mathbf{m}_{ij}$ and covariance matrix Σ. As a consequence, every class $c_i, i = 1, \ldots, L$ can be represented as a mixture of Gaussians with probability density function

$$p(\mathbf{x} \mid c_i) = \sum_{j=1}^{n_i} q_{ij} \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{m}_{ij})^T\Sigma^{-1}(\mathbf{x} - \mathbf{m}_{ij})\right) \tag{15}$$

$$q_{ij} = \frac{\pi_{ij}}{(2\pi)^{D/2}|\Sigma|^{1/2}}. \tag{16}$$

Applying the Bayes rule, and making some straightforward algebraic manipulations, it is easy to check that the posterior probability of class i is

$$P(c_i \mid \mathbf{x}) = \frac{P(c_i)p(\mathbf{x} \mid c_i)}{\sum_{k=1}^{L} P(c_k)p(\mathbf{x} \mid c_k)} = \frac{\sum_{j=1}^{n_i} \exp\left(\mathbf{w}_{ij}^T \mathbf{x} + b_{ij}\right)}{\sum_{s=1}^{L} \sum_{t=1}^{n_s} \exp\left(\mathbf{w}_{st}^T \mathbf{x} + b_{st}\right)} \tag{17}$$

where $P(c_i)$ is the *a priori* probability for class i

$$\mathbf{w}_{ij} = \mathbf{m}_{ij}^T \Sigma^{-1} \tag{18}$$

and

$$b_{ij} = \ln(P(c_i)\pi_{ij}) - \frac{1}{2}\mathbf{m}_{ij}^T \Sigma^{-1} \mathbf{m}_{ij}. \tag{19}$$

Therefore, if $\mathbf{g}(\cdot)$ is the softmax function, we can write

$$P(c_i \mid \mathbf{x}) = \sum_{j=1}^{n_i} g_{ij}(\mathbf{W}^T \mathbf{x}_\mathbf{e}) \tag{20}$$

where matrix $\mathbf{W}$ encompasses all weight vectors and biases in (18) and (19), and $\mathbf{x}_\mathbf{e}^T = (\mathbf{x}^T\ 1)$. This proves that the softmax based GSP can compute any posterior probability map for data coming from Gaussian mixture models, having all the gaussians the same covariance matrix Σ. Since any distribution can be approximated as a Gaussian mixture [9], [10], the generalized softmax perceptron is a universal approximator.[1]

D. Learning

Consider now the following learning problem: let $\mathcal{S}$ be a sample set made up of observations and their labels: $\mathcal{S} = \{(\mathbf{x}^{(k)}, \mathbf{d}^{(k)}), k = 1, \ldots, K\}$, where their elements have been generated independently according to an unknown joint probability function $p(\mathbf{x}, \mathbf{d})$, where $\mathbf{x}$ is an element of an observation space $\mathcal{X} \subset \mathbb{R}^D$ and $\mathbf{d} \in \mathcal{U}_L = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_L\}$. Class $\mathbf{u}_i = (0, \ldots, 1, \ldots, 0)$ is a vector with a unique "1" in the position corresponding to u_i.

We are interested here in learning rules obtained for stochastic gradient minimization of a cost function $C(\mathbf{y}, \mathbf{d})$

$$\mathbf{w}_{ij}(k+1) = \mathbf{w}_{ij}(k) - \rho \nabla_{\mathbf{w}_{ij}} C \tag{21}$$

where ∇ is the gradient operator.

In general, the form of the learning rule depends on the cost function and the activation function, $\mathbf{g}$. A particular interesting case is found when $\mathbf{g}$ is the softmax nonlinearity and C is the cross entropy

$$C(\mathbf{y}, \mathbf{d}) = -\sum_{i=1}^{L} d_i \log y_i. \tag{22}$$

It is not difficult to show that the stochastic gradient learning rule becomes

$$\mathbf{w}_{ij}(k+1) = \mathbf{w}_{ij}(k) + \rho \frac{y_{ij}}{y_i}(d_i - y_i)\mathbf{x}. \tag{23}$$

There are several advantages of this rule.

- It is simple and easy to interpret: for weight vectors $\mathbf{w}_{i,1}, \ldots, \mathbf{w}_{i,n_i}$, the correction term is proportional to the error in class i (i.e., $d_i - y_i$), but the error is shared among these weights depending on the value of y_{ij}/y_i. This shows that learning is supervised at the class level but unsupervised and competitive among weight vectors with the same class index.
- The rule leads to posterior probability estimates. This is because it comes from the minimization of the cross entropy, which is known to have this property.
- It is a direct extension of that in (1), which is known to be well-formed [21], for $N \geq L \geq 2$.

The main goal of this paper is to analyze the possibility of using learning rule (23) when $\mathbf{g}$ is not (necessarily) equal to the softmax and C is not (necessarily) the cross entropy. To do so, we proceed as follows.

1) We find necessary and sufficient conditions on the cost function and the activation function to guarantee that this learning rule provides posterior probability estimates.
2) We generalize the concept of well-formed cost functions to multiclass problems.
3) We find conditions on the activation function so that the learning rule minimizes a well-formed cost function.

III. Fundamentals: A Review on SSB Cost Functions

The cost functions providing posterior probability estimates in multiclass problems have been discussed in [4] and [5], where they are called strict sense Bayesian (SSB). We take one result from these papers that will be useful to prove the universality of learning rule (23).

Definition 1) SSB Cost Function: Cost function $C : \mathcal{P}_L \times \mathcal{U}_L \rightarrow \mathbb{R}$, is said to be SSB if $E\{C(\mathbf{y}, \mathbf{d}) \mid \mathbf{x}\}$ has a unique minimum when $\mathbf{y}$ is the posterior class probability vector for every $\mathbf{x} \in \mathcal{X}$, i.e.

$$y_i = P(\mathbf{d} = \mathbf{u}_i \mid \mathbf{x}), \quad i = 1, \ldots, L. \tag{24}$$

A general expression for SSB cost functions is given by the following theorem [4].

Theorem 1: A cost function $C : \mathcal{P}_L \times \mathcal{U}_L \rightarrow \mathbb{R}$, is SSB iff it can be written in the form

$$C(\mathbf{y}, \mathbf{d}) = h(\mathbf{y}) + (\mathbf{d} - \mathbf{y})^T \nabla_\mathbf{y} h(\mathbf{y}) \tag{25}$$

where $h : \mathcal{P}_L \rightarrow \mathbb{R}$ is a strictly convex function (in $\mathcal{P}_L$).

In the next section, the learning rule corresponding to the GSP is stated, deriving conditions ensuring that the output of the trained network approximates the conditional expectation of the desired output.

IV. Universality Conditions

In this section, and once all factors involved in the classification scheme have been presented, we state and prove the universality conditions.

First of all, we define the concept of conditional separability, which is a property of an activation function in the context of a particular GSP structure.

[1]The same conclusion could have been achieved if the nonparametric kernel estimator provided by Nadaraya–Watson [15], [20] had been used.

Definition 2) Conditional Separability: Probabilistic activation function $\mathbf{g}$ of a GSP is said to be conditionally separable if variables

$$z_{j\,|\,i} = \frac{y_{ij}}{y_i} \tag{26}$$

with $0 < i \leq L$, do not depend on o_{mn}, with $m \neq i$, i.e.

$$\frac{\partial z_{j\,|\,i}}{\partial o_{mn}} = 0. \tag{27}$$

Note that, if $\mathbf{g}$ is conditionally separable, each variable $z_{j\,|\,i}$ can be expressed as a function of $\mathbf{o_i} = (o_{i1}, \ldots, o_{in_i})$. In the following, such functions are named conditional activations.

Definition 3) Conditional Activation Function: Given a GSP with probabilistic activation function $\mathbf{g}$, function $\mathbf{g}_i : \mathbb{R}^{n_i} \rightarrow \mathcal{P}_{n_i}$ is said to be the ith conditional activation of the GSP if $\mathbf{z}_i = (z_{1\,|\,i}, \ldots, z_{n_i\,|\,i})$ can be expressed as

$$\mathbf{z}_i = \mathbf{g}_i(\mathbf{o_i}). \tag{28}$$

The softmax nonlinearity $\mathbf{g}$ given by (14) is an example of conditionally separable function. To show it, note that

$$z_{j\,|\,i} = \frac{y_{ij}}{y_i} = \frac{\exp(o_{ij})}{\sum_{t=1}^{n_i} \exp(o_{it})} \tag{29}$$

does not depend on any o_{mn} such that $m \neq i$. Its ith conditional activation function, $\mathbf{g}_i$, is a vector where its jth element is given by (29).

As the following theorems show, the conditional separability is essential to guarantee that learning rule (23) provides posterior probability estimates. First, we provide some necessary conditions.

Theorem 2: Consider a L class problem with $L > 2$ and let $\mathbf{Y}$ be the output of a GSP with probabilistic activation function $\mathbf{g}$. If the learning rule

$$\mathbf{w}_{ij}(k+1) = \mathbf{w}_{ij}(k) + \rho \frac{y_{ij}}{y_i}(d_i - y_i)\mathbf{x} \tag{30}$$

minimizes a SSB cost function $C : \mathcal{P}_L \times \mathcal{U}_L \rightarrow \mathbb{R}$, then the following propositions are true.

1) $\mathbf{g}$ is the gradient of a potential function, i.e., there exists $f(\mathbf{o})$ such that $\mathbf{Y} = \mathbf{g}(\mathbf{o}) = \nabla_{\mathbf{o}} f$.
2) $\mathbf{g}$ is conditionally separable
3) Conditional activations $\mathbf{g}_i$ are gradients of potential functions, i.e., there exists functions $f_i : \mathbb{R}^{n_i} \rightarrow \mathcal{P}_{n_i}$ such that

$$\mathbf{z}_i = \nabla_{\mathbf{o}_i} f_i(\mathbf{o_i}). \tag{31}$$

4) Matrix $\mathbf{P_F}$, defined as $\mathbf{P_F} = \mathbf{UF}$, where $\mathbf{U}$ is given by (6) and $\mathbf{F}$ is the Hessian matrix of f, satisfies that $\text{rank}(\mathbf{P_F}) = L - 1$.
5) Matrix $\mathbf{X} = \mathbf{P_F^T J}$, where

$$\mathbf{J} = \begin{pmatrix} \mathbf{z}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{z}_2 & \cdots & \mathbf{0} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{z}_L \end{pmatrix} \tag{32}$$

is semidefinite–positive in $\mathbb{R}^N$.

Proof: See the Appendix.

An important question is if the set of Propositions 1–5 in Theorem 2 is also sufficient. Next theorem is proposed to answer it.■

Theorem 3: Consider a L class problem with $L > 2$ and let $\mathbf{Y}$ be the output of a GSP with probabilistic ativation $\mathbf{g}$. Assume the following propositions are true.

1) Function $\mathbf{g}$ is the gradient of a potential function, i.e., there exists $f(\mathbf{o})$ such that $\mathbf{Y} = \mathbf{g}(\mathbf{o}) = \nabla_{\mathbf{o}} f$.
2) $\mathbf{g}$ is conditionally separable
3) Conditional activations $\mathbf{g}_i$ are gradients of potential functions, i.e., there exists functions $f_i : \mathbb{R}^{n_i} \rightarrow \mathcal{P}_{n_i}$ such that

$$\mathbf{z}_i = \nabla_{\mathbf{o}_i} f_i(\mathbf{o_i}). \tag{33}$$

4) Matrix $\mathbf{X}$ given by

$$\mathbf{X} = \mathbf{F}^T \mathbf{U}^T \mathbf{J} \tag{34}$$

where J is given by (32), $\mathbf{F}$ is the Hessian matrix of f and U is given by (6), is semidefinite–positive in $\mathbb{R}^N$.

Learning rule

$$\mathbf{w}_{ij}(k+1) = \mathbf{w}_{ij}(k) + \rho \frac{y_{ij}}{y_i}(d_i - y_i)\mathbf{x} \tag{35}$$

minimizes a cost function $C : \mathbb{R}^N \times \mathcal{U}_L \rightarrow \mathbb{R}$ that leads to posterior probability estimates.

Proof: See the Appendix. ■

Note that the theorem ensures that learning rule (35) leads to posterior probability estimates, although this does not mean that cost function C is SSB, because, in general, C is function of $\mathbf{Y}$ and $\mathbf{d}$, but in some cases it could be impossible to express it as a function of $\mathbf{y}$ and $\mathbf{d}$, which is a condition to be SSB, according to Definition 1. Disregarding this minor aspect, we can say that Propositions 1 to 4 state, essentially an "iff" condition.

It is noteworthy that the theorem provides an expression to state the cost minimized by (35) using the potential functions [see (106)] as

$$C(\mathbf{o}, \mathbf{d}) = f(\mathbf{o}) - \sum_{k=1}^{L} d_k f_k(\mathbf{o}_k) \tag{36}$$

Theorem 2 applies to a specific assignment of the activation outputs to different classes. The following theorem may be of interest to apply the same activation for different assignments.

Theorem 4: Let $\mathbf{Y}$ be the output of a GSP with nonlinearity $\mathbf{g}$. If learning rule

$$\mathbf{w}_{ij}(k+1) = \mathbf{w}_{ij}(k) + \rho \frac{y_{ij}}{y_i}(d_i - y_i)\mathbf{x} \tag{37}$$

minimizes a cost function $C : \mathbb{R}^N \times \mathcal{U}_L \rightarrow \mathbb{R}$ that leads to posterior probability estimates for any number of classes L and any partition $N = \sum_{i=1}^{L} n_i$, then the following statements are satisfied.

1) Function $\mathbf{g}$ is the gradient of a potential function, i.e., there exists $f(\mathbf{o})$ such that $\mathbf{Y} = \mathbf{g}(\mathbf{o}) = \nabla_{\mathbf{o}} f$.
2) Hessian matrix $\mathbf{F}$, of function f satisfies that $\text{rank}(\mathbf{F}) = N - 1$.

3) Matrix $\mathbf{F}$ is semidefinite–positive in $\mathbb{R}^N$ (i.e., f is concave).

The proof of the theorem results from analyzing Theorem 2 for $n_i = 1$.

V. Well-Formed Cost Functions

Although (23) is a natural extension of (1), we need some theoretical results showing that it has also a good behavior. To do so, in this section we extend the concept of well-formed cost function [21] to multiclass problems. Our analysis here is restricted to GSP structures with a single linear combination per class (i.e., $n_i = 1$ for all i). The extension of this result to the general case is a matter of our current research.

The extension is based on the concept of "margin" of a vector, that we define in the following.

Definition 4) Margin of a Vector: Given vector $\mathbf{o} \in \mathbb{R}^N$, and $m = \arg\max_i\{o_i\}$, the margin of $\mathbf{o}$ is denoted as margin($\mathbf{o}$) and defined as

$$\text{margin}(\mathbf{o}) = o_m - \max_{i \neq m}\{o_i\}. \tag{38}$$

Now, we will define the concept of well-formed cost functions. We should make clear beforehand that the definition is oriented to GSP-like networks with $n_i = 1$ (i.e., a single filter per class), activation function satisfying property P3 in Section II-B, and WTA decision, so that we can write $\hat{d}_j = 1$ if $j = \arg\max_i\{o_i\}$. and, therefore, if $\mathbf{d}$ is the target vector $\mathbf{o}^T\mathbf{d} \neq \max_i\{o_i\}$ implies that there is a misclassification error.

Definition 5) Well-Formed Cost Function: Cost function $C : \mathbb{R}^L \times \mathcal{U}_L \to \mathbb{R}$ is said to be well-formed if the following are true.

1) For any $\mathbf{o}^* \in \mathbb{R}^N$ such that $\mathbf{o}^{*T}\mathbf{d} = \max_i\{o_i^*\}$
$$-(\nabla_{\mathbf{o}} C(\mathbf{o}, \mathbf{d}))^T \mathbf{o}^* \geq 0 \tag{39}$$
(i.e., C never pushes in the wrong direction).
2) There exists $\varepsilon > 0$ such that, for any $\mathbf{o} \in \mathbb{R}^N$ such that $\mathbf{o}^T\mathbf{d} \neq \max_i\{o_i\}$ and any $\mathbf{o}^* \in \mathbb{R}^N$ such that $\mathbf{o}^{*T}\mathbf{d} = \max_i\{o_i^*\}$,
$$-(\nabla_{\mathbf{o}} C(\mathbf{o}, \mathbf{d}))^T \mathbf{o}^* \geq \varepsilon \text{margin}(\mathbf{o}^*) \tag{40}$$
(i.e., C keeps pushing if there is a misclassification).
3) C is bounded below.

Note that this definition extends that proposed in [21], in the sense that both of them are equivalent for $L = 2$.

We will now prove that, provided that data are separable, well-formed cost functions are guaranteed to find separating boundaries.

Theorem 5: Consider a GSP with L classes, $n_i = 1\ \forall i$ and activation function satisfying property P3 in Section II-B. Also, consider gradient descent rule with differential step size given by

$$\frac{d\mathbf{w}}{dt} = -\mu \nabla_{\mathbf{w}} E(\mathbf{w}) \tag{41}$$

where $\mu > 0$ and E is the cumulative cost over the training set

$$E(\mathbf{w}) = \sum_{k=1}^{K} C(\mathbf{o}^k, \mathbf{d}^k). \tag{42}$$

If C is well-formed, the learning rule converges to a coefficient matrix $\mathbf{w}^*$ with zero errors provided the data are separable.

Proof: Note that

$$\begin{aligned}\frac{dE}{dt} &= \sum_{i=1}^{L}\sum_{j=1}^{n_i} \frac{\partial E}{\partial w_{ij}} \frac{dw_{ij}}{dt} \\ &= -\frac{1}{\mu}\sum_{i=1}^{L}\sum_{j=1}^{n_i} \frac{dw_{ij}}{dt}\frac{dw_{ij}}{dt} \\ &= -\frac{1}{\mu}\left\|\frac{d\mathbf{w}}{dt}\right\|^2. \end{aligned} \tag{43}$$

On the other hand

$$\begin{aligned}\frac{d\mathbf{w}_i}{dt} &= -\mu\nabla_{\mathbf{w}_i} E = -\mu\sum_{k=1}^{K} \nabla_{\mathbf{w}_i} C \\ &= -\mu\sum_{k=1}^{K}\sum_{j=1}^{L} \frac{\partial C}{\partial o_j^k}\nabla_{\mathbf{w}_i} o_j^k \\ &= -\mu\sum_{k=1}^{K} \frac{\partial C}{\partial o_i^k}\mathbf{x}^k. \end{aligned} \tag{44}$$

Let $\mathbf{w}^*$ be a separating boundary and let

$$\lambda = \min_k\{\text{margin}(\mathbf{o}^{*k}), k = 1, \ldots, K\} \tag{45}$$

then

$$\begin{aligned}\sum_{i=1}^{L} \mathbf{w}_i^{*T}\frac{d\mathbf{w}_i}{dt} &= -\mu\sum_{i=1}^{L}\sum_{k=1}^{K} \frac{\partial C}{\partial o_i^k}\mathbf{w}_i^{*T}\mathbf{x}^k \\ &= -\mu\sum_{k=1}^{K}\sum_{i=1}^{L} \frac{\partial C}{\partial o_i^k} o_i^{*k}. \end{aligned} \tag{46}$$

Using Property 1 (39), we can lower bound the previous expression as

$$\sum_{i=1}^{L} \mathbf{w}_i^{*T}\frac{d\mathbf{w}_i}{dt} \geq -\mu \sum_{\mathbf{x}^k \in \mathcal{S}_u}\sum_{i=1}^{L} \frac{\partial C}{\partial o_i^k} o_i^{*k} \tag{47}$$

where $\mathcal{S}_u$ is the set of all samples incorrectly classified. Also, using Property 2 (40), we get

$$\sum_{i=1}^{L} \mathbf{w}_i^{*T}\frac{d\mathbf{w}_i}{dt} \geq \varepsilon\mu \sum_{\mathbf{x}^k \in \mathcal{S}_u} \text{margin}(\mathbf{o}^{*k}). \tag{48}$$

Using (45), we get

$$\sum_{i=1}^{L} \mathbf{w}_i^{*T}\frac{d\mathbf{w}_i}{dt} \geq n_e\varepsilon\lambda\mu \tag{49}$$

where n_e is the number of errors. Since $\mathbf{w}_i^{*T}(d\mathbf{w}_i)/(dt) \leq \|\mathbf{w}_i^*\|\|(d\mathbf{w}_i)/(dt)\|$, we can bound (43) as

$$\frac{dE}{dt} = -\frac{1}{\mu}\left\|\frac{d\mathbf{w}}{dt}\right\|^2 \leq -\mu\left(\frac{n_e\varepsilon\lambda}{\|\mathbf{w}^*\|}\right)^2. \tag{50}$$

Thus, if the algorithm converges, it cannot converge to a vector with $n_e \neq 0$ (in such a case, E would decrease to $-\infty$, which is in contradiction with the fact that C is bounded below). ■

For $n_i = 1$, learning rule (23) becomes

$$\mathbf{w}_i(k+1) = \mathbf{w}_i(k) + \rho(d_i - y_i)\mathbf{x}. \tag{51}$$

The following theorems shows that, for a wide family of probabilistic activation functions, rule (51) minimizes a well-formed cost function.

Theorem 6: Consider a GSP with L classes, $n_i = 1$ for all i, and probabilistic activation function $\mathbf{g}$ satisfying property P3 in Section II-B. If:

1) $\mathbf{g}$ is the gradient of a potential function, i.e., there exists $f(\mathbf{o})$ such that $\mathbf{y} = \mathbf{g}(\mathbf{o}) = \nabla_{\mathbf{o}} f$;
2) for every $\mathbf{o}$, $f(\mathbf{o}) \geq o_i, \forall i$;
3) the Hessian matrix of $f, \mathbf{F}$, is positive–semidefinite and satisfies that $\text{rank}(\mathbf{F}) = L - 1$.

Then, learning rule (51) is the stochastic gradient rule minimizing a cost function $C(\mathbf{y}, \mathbf{d})$ that satisfies the following properties:

1) C is SSB;
2) C is well-formed.

Proof: The proof that (51) minimizes an SSB cost can be found in [4, Th. 5] and is also a particular of Theorem 3 in this paper. To prove that C is well-formed, note that

$$\frac{\partial C}{\partial o_i} = (y_i - d_i). \tag{52}$$

Therefore, for any $\mathbf{o}^* \in \mathbb{R}^N$ such that $\mathbf{d}^T\mathbf{o}^* = \max_i\{o_i^*\} = o_m^*$

$$\begin{aligned} -(\nabla_{\mathbf{o}} C(\mathbf{o}, \mathbf{d}))^T \mathbf{o}^* &= (\mathbf{d} - \mathbf{y})^T \mathbf{o}^* \\ &= \max_i\{o_i^*\} - \mathbf{y}^T\mathbf{o}^* \\ &= o_m^* - \mathbf{y}^T\mathbf{o}^*. \end{aligned} \tag{53}$$

Since $\mathbf{y}^T\mathbf{o}^*$ is a weighted average of the elements in vector $\mathbf{o}^*$, we can write

$$o_m^* \geq \mathbf{y}^T\mathbf{o}^* \tag{54}$$

so that

$$-(\nabla_{\mathbf{o}} C(\mathbf{o}, \mathbf{d}))^T \mathbf{o}^* \geq 0 \tag{55}$$

what proves that C never pushes in the wrong direction.

Also, note that, for any $\mathbf{o} \in \mathbb{R}^N$ such that $\mathbf{o}^T\mathbf{d} \neq \max_i\{o_i\}$ and any $\mathbf{o}^* \in \mathbb{R}^N$ such that $\mathbf{o}^{*T}\mathbf{d} = \max_i\{o_i^*\} = o_m^*$

$$\begin{aligned} -(\nabla_{\mathbf{o}} C(\mathbf{o}, \mathbf{d}))^T \mathbf{o}^* &= o_m^* - \sum_{i=1}^{L} y_i o_i^* \\ &= (1 - y_m) o_m^* - \sum_{i \neq m} y_i o_i^* \\ &= \sum_{i \neq m} y_i (o_m^* - o_i^*) \\ &\geq y_j (o_m^* - o_j^*), \quad \forall j \neq m \end{aligned} \tag{56}$$

where $\mathbf{o}^T\mathbf{d} = o_j$. The last inequality holds because all terms in the sum are positive numbers. Specifically, for index j corresponding to $y_j = \max_i\{y_i\}$, we have that $y_j \geq 1/L$ and $(o_m^* - o_j^*) \geq \text{margin}(\mathbf{o}^*)$, bounding (56) by

$$-(\nabla_{\mathbf{o}} C_h(\mathbf{o}, \mathbf{d}))^T \mathbf{o}^* \geq \frac{1}{L} \text{margin}(\mathbf{o}^*) \tag{57}$$

what proves that C keeps pushing if there is a misclassification.

To prove that C is bounded below, note first that, according to Condition 2 of this theorem

$$f(\mathbf{o}) \geq o_i, \quad \forall i. \tag{58}$$

Besides, if the classifier has a single filter per class, $n_i = 1$, we have that $z_{j\,|\,i} = y_{ij}/y_i = 1$ which, replaced in (31), leads to

$$\nabla_{o_i} f_i(o_i) = 1 \Rightarrow f_i(o_i) = o_i. \tag{59}$$

Combination of (58), (59), and (36) leads to conclude that C is bounded below. ■

In the following we present two examples to show that not all SSB cost functions are well-formed in the above sense. In particular, we examine the case of the sum-squared error (SSE) cost and the cross entropy when the softmax activation function is used.

- The SSE cost function can be expressed as

$$C_{\text{sse}}(\mathbf{o}, \mathbf{d}) = \sum_{i=1}^{L} (d_i - y_i)^2 = \sum_{i=1}^{L} e_i^2$$

where the dependence with $\mathbf{o}$ is represented through vector $\mathbf{y}$. Now, we show that Condition 2 in Definition 5 is not satisfied. The gradient vector $\nabla_{\mathbf{o}} C_{\text{sse}}$ is given by components

$$\frac{\partial C_{\text{sse}}(\mathbf{o}, \mathbf{d})}{\partial o_i} = -2y_i \left[e_i - \sum_{j=1}^{L} y_j e_j \right], \quad i = 1, \ldots, L$$

and, assuming that m is the right class, the left-hand side of (40) takes the form

$$\begin{aligned} -(\nabla_{\mathbf{o}} C_{\text{sse}}(\mathbf{o}, \mathbf{d}))^T \mathbf{o}^* &= 2y_m(1 - y_m)(o_m^* - \mathbf{o}^{*T}\mathbf{y}) \\ &\quad + 2 \sum_{\substack{i=1 \\ i \neq m}}^{L} y_i^2 (\mathbf{o}^{*T}\mathbf{y} - o_i^*). \end{aligned} \tag{60}$$

Since the softmax nonlinearity satisfies Property P4 in Section II-B, and if $o_m^* \to \infty$, then

$$y_m \to 1 \quad \text{and therefore} \quad \mathbf{o}^{*T}\mathbf{y} \to o_m^* \tag{61}$$

so that, for o_m^* large enough

$$(o_m^* - \mathbf{o}^{*T}\mathbf{y}) < \text{margin}(\mathbf{o}). \tag{62}$$

Using (61) and (62), (60) can be bounded by

$$\begin{aligned} -(\nabla_{\mathbf{o}} C_{\text{sse}}(\mathbf{o}, \mathbf{d}))^T \mathbf{o}^* &\leq 2y_m(1 - y_m)\, \text{margin}(\mathbf{o}) \\ &\quad + 2 \sum_{\substack{i=1 \\ i \neq m}}^{L} y_i^2 (o_m^* - o_i^*) \end{aligned} \tag{63}$$

and taking into account that $(o_m^* - o_i^*) < \text{margin}(\mathbf{o})$, we can write (63) as follows:

$$-(\nabla_{\mathbf{o}} C_{\text{sse}}(\mathbf{o}, \mathbf{d}))^T \mathbf{o}^* \leq 2K \text{margin}(\mathbf{o}) \tag{64}$$

where $K = y_m(1 - y_m) + \sum_{\substack{i=1 \\ i \neq m}}^{L} y_i^2$ just takes positive values. On the other hand, since $y_m \to 1$ (61), then $K \to 0$ and Condition 2 in Definition 5 cannot be satisfied for any constant $\epsilon > 0$.

- Now we demonstrate that a softmax nonlinearity with the cross-entropy function presented in (22) is well-formed. To do so, note, first, that (51) is the stochastic gradient learning rule for this case. Therefore, we will prove the well-formed property by showing that conditions in Theorem 6 are satisfied.

  Regarding the first condition, it is straightforward to verify that

$$f(\mathbf{o}) = \log\left(\sum_{j=1}^{L} \exp(o_j)\right) \tag{65}$$

  is a potential function for the softmax. To check the second requisite, we start from the following inequality:

$$\sum_{j=1}^{L} \exp(o_j) \geq \exp(o_i), \quad \forall i \tag{66}$$

  and apply the logarithm function to both sides of (66)

$$\log\left(\sum_{j=1}^{L} \exp(o_j)\right) \geq o_i, \quad \forall i. \tag{67}$$

  Since the expression on the left-hand side of (67) is identical to $f(\mathbf{o})$, we conclude that $f(\mathbf{o}) \geq o_i, \forall i$.

  The last condition of Theorem 6 requires to verify that the Hessian matrix of f, which is given by

$$\mathbf{F} = \operatorname{diag}(\mathbf{y}) - \mathbf{y}\mathbf{y}^T \tag{68}$$

  (where $\operatorname{diag}(\mathbf{y})$ refers to a square diagonal matrix with the elements of $\mathbf{y}$ on the main diagonal) is positive–semidefinite with $\operatorname{rank}(\mathbf{F}) = \mathbf{L - 1}$. To prove that $\mathbf{F}$ is positive–semidefinite, note that, for any $\mathbf{a} \in \mathbb{R}^L$

$$\mathbf{a}^T\mathbf{F}\mathbf{a} = \sum_{i=1}^{L} a_i^2 y_i - \left(\sum_{i=1}^{L} a_i y_i\right)^2 \tag{69}$$

  and, after some rearrangement, takes the form

$$\mathbf{a}^T\mathbf{F}\mathbf{a} = \sum_{i=1}^{L} y_i \left(a_i - \sum_{j=1}^{L} a_j y_j\right)^2 \geq 0 \tag{70}$$

  proving that $\mathbf{F}$ is positive–semidefinite.

  To determine $\operatorname{rank}(\mathbf{F})$ we compute the null space of $\mathbf{F}$ ($\mathcal{N}(\mathbf{F})$), i.e., the solutions of

$$\mathbf{F}\mathbf{a} = \mathbf{0}, \quad \forall \mathbf{a} \neq \mathbf{0} \tag{71}$$

  under the constraint that $\sum_{i=1}^{L} y_i = 1$. It is easy to check that only vectors with the form $\mathbf{a} = k\mathbf{1}$, where $\mathbf{1}$ is the vector with all components equal to 1 and k is any constant value, are a solution of (71). This implies that $\dim\mathcal{N}(\mathbf{F}) = 1$ and therefore $\operatorname{rank}(\mathbf{F}) = L - 1$.

VI. Experiment

As an example, we compare in this section the difference in the posterior class probabilities estimates provided by a probabilistic neural network (PNN) [17] and our GSP scheme. To evaluate the quality of the different estimates we have turned to the average Kullback–Leibler (KL) divergence D_{kl} between the true and estimated posterior probabilities, respectively, denoted by $P(c_i \,|\, \mathbf{x})$ and $\hat{P}(c_i \,|\, \mathbf{x})$:

$$D_{kl} = \frac{-1}{S} \sum_{j=1}^{S} \sum_{i=1}^{L} P\left(c_i \left| \mathbf{x}^{(j)} \right.\right) \log \frac{\hat{P}\left(c_i \left| \mathbf{x}^{(j)} \right.\right)}{P\left(c_i \left| \mathbf{x}^{(j)} \right.\right)}$$

where S is the number of samples to consider.

The PNN presented in [17] is based on a nonparametric kernel estimation of class densities, i.e., $\hat{p}(\mathbf{x} \,|\, c_i), i = 1, \ldots, L$. In our experiment, we will use a Gaussian kernel where the smoothing parameter σ is chosen to minimize the following cost function:

$$C_{\text{pnn}} = -\sum_{j=1}^{K} \sum_{i=1}^{L} d_i^{(j)} \log \hat{P}\left(c_i \,\middle|\, \mathbf{x}^{(j)}\right)$$

where K is the number of training samples and $\log \hat{P}(c_i \,|\, \mathbf{x}) = \hat{p}(\mathbf{x} \,|\, c_i) / \sum_{c=1}^{L} \hat{p}(\mathbf{x} \,|\, c_c)$, since all classes have the same prior probability. Parameter σ is found through a leave-one-out procedure.

To train the GSP network we have used the cross-entropy SSB cost function given in (22) together with the softmax nonlinearity. The training procedure follows a stochastic gradient descent method (learning rule (23)) where the learning step ρ decreases according to $\rho_k = \rho_0/(1 + k/k_0)$, where k is the iteration number and ρ_0 and k_0 are set to 1 and 1000, respectively. The number of epochs have been fixed to 100. Two architectures have been regarded for the GSP network, differing on the number of filters per class. Specifically, we have experienced with one and two filters per class.

To illustrate the difference between PNN and our proposal, consider a synthetic two-dimensional classification problem with three classes ($L = 3$) of equal prior probabilities. Each class is a mixture of two Gaussian distributions with the same prior probability and variance $v = 0.005$. The mean vectors for every Gaussian are: $\mathbf{m}_{11} = \{1/3, 0.7\}, \mathbf{m}_{12} = \{0.25, 0.2\}$ for class C_1; $\mathbf{m}_{21} = \{0.5, 0.25\}, \mathbf{m}_{22} = \{0.6, 0.6\}$ for C_2; and $\mathbf{m}_{31} = \{0.3, 0.4\}, \mathbf{m}_{32} = \{0.7, 0.4\}$ for C_3.

The dependence of the posterior class estimate with the size of the training set has also been taken into account in our simulations. Specifically, we have generated five training sets, consisting of 2, 20, 100, 200, and 400 samples per Gauss. To evaluate the performance an independent test set with 800 instances/class has been created.

Fig. 2 displays the D_{kl} for the test set and the three schemes: PNN and GSP with one and two "subclasses." Owing to the dependence of the GSP solution with the network initialization, we have represented here the average over 10 runs. As expected, the GSP with one filter per class provides the worst estimate (higher D_{kl} in all cases): just one filter per class is unable to model data coming from a mixture of Gaussians. Note how the increase in the GSP complexity (2 filters/class) significantly reduces the KL-divergence (solid line), outperforming the PNN method in all cases.

VII. Conclusion

This paper analyzes a simple stochastic gradient learning rule for GSP networks. We provide theoretical results indicating the

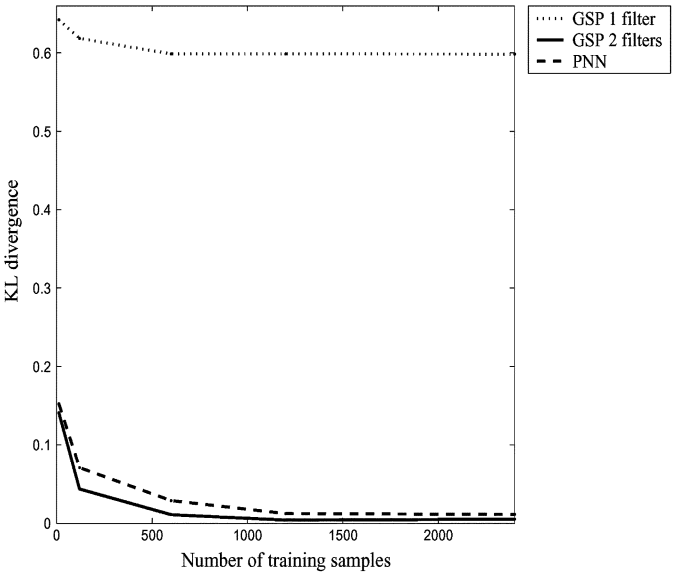

Fig. 2. KL divergence versus number of training samples. Dotted line: GSP with one filter/class. Solid line : GSP with two filters/class. Dashed line: PNN.

learning rule provides estimates of posterior class probabilities, under some conditions on the activation function of the network. Also, for multiclass problems with a single filter per class, we show that the learning rule is well-formed.

It is possible to apply the universal learning rule and the kind of cost functions we have studied using a variety of nonlinear transfer functions. This means that it is not necessary to know the shape of the nonlinearity with great accuracy. We guess this may be interesting for analog implementations of this kind of networks [11], [22]. Since electronic components have a certain tolerance, the precise shape of activation functions may be out of the designer control.

Although the theoretical results can be trivially extended to universal classifiers based on the use of nonlinear maps over the input space, the extension of the well-formed concept to general GSP architectures with arbitrary approximation capability is an open issue, that is being currently investigated by the authors.

APPENDIX

In this section, we will prove the theorems presented in Section IV.

A. Proof of Theorem 2

Proof of Proposition 1: Let us assume that (30) minimizes cost function C. Then, we can also express $\mathbf{w}_{ij}(k+1)$ as

$$\begin{aligned}\mathbf{w}_{ij}(k+1) &= \mathbf{w}_{ij}(k) - \rho\nabla_{\mathbf{w}_{ij}}C \\ &= \mathbf{w}_{ij}(k) - \rho\frac{\partial C}{\partial o_{ij}}\nabla_{\mathbf{w}_{ij}}o_{ij} \\ &= \mathbf{w}_{ij}(k) - \rho\frac{\partial C}{\partial o_{ij}}\mathbf{x}. \end{aligned} \tag{72}$$

Comparing (30) and (72) we obtain

$$\frac{\partial C}{\partial o_{ij}} = \frac{y_{ij}}{y_i}(y_i - d_i). \tag{73}$$

Expressing the cost function as

$$C(\mathbf{y},\mathbf{d}) = \sum_{k=1}^{L} d_k c_k(\mathbf{y}) \tag{74}$$

where $c_k(\mathbf{y}) = C(\mathbf{y},\mathbf{u}_k)$ and $\mathbf{u}_k \in \mathcal{U}_L$, we get

$$\frac{\partial C(\mathbf{y},\mathbf{d})}{\partial o_{ij}} = \sum_{k=1}^{L} d_k \frac{\partial c_k(\mathbf{y})}{\partial o_{ij}}. \tag{75}$$

Using (73) and (75) we have

$$\sum_{k=1}^{L} d_k \frac{\partial c_k(\mathbf{y})}{\partial o_{ij}} = \frac{y_{ij}}{y_i}(y_i - d_i) = z_{j\,|\,i}(y_i - d_i). \tag{76}$$

The previous equality must hold for every target vector $\mathbf{d}$. For instance, if $\mathbf{d} = \mathbf{u}_p$ we get

$$\frac{\partial c_p(\mathbf{y})}{\partial o_{ij}} = \frac{y_{ij}}{y_i}(y_i - \delta_{i-p}). \tag{77}$$

Differentiating both sides of (77) with respect to o_{st}, we obtain

$$\frac{\partial^2 c_p(\mathbf{y})}{\partial o_{st}\partial o_{ij}} = \frac{\partial y_{ij}}{\partial o_{st}} - \delta_{i-p}\frac{\partial}{\partial o_{st}}\left(\frac{y_{ij}}{y_i}\right). \tag{78}$$

If we interchange the derivation order, we have

$$\frac{\partial^2 c_p(\mathbf{y})}{\partial o_{ij}\partial o_{st}} = \frac{\partial y_{st}}{\partial o_{ij}} - \delta_{s-p}\frac{\partial}{\partial o_{ij}}\left(\frac{y_{st}}{y_s}\right). \tag{79}$$

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

$$\frac{\partial y_{ij}}{\partial o_{st}} - \delta_{i-p}\frac{\partial}{\partial o_{st}}\left(\frac{y_{ij}}{y_i}\right) = \frac{\partial y_{st}}{\partial o_{ij}} - \delta_{s-p}\frac{\partial}{\partial o_{ij}}\left(\frac{y_{st}}{y_s}\right). \tag{80}$$

Note that this equation holds for every value of s,t,i,j,p. Therefore, considering $p \neq i$ and $p \neq s$ we can write the following equality, which holds no matter what is the value of p:

$$\frac{\partial y_{ij}}{\partial o_{st}} = \frac{\partial y_{st}}{\partial o_{ij}}. \tag{81}$$

From (81) we conclude that $\mathbf{Y} = \mathbf{g}(\mathbf{o})$ is the gradient of a potential function f. That is

$$\mathbf{Y} = \boldsymbol{\nabla}_{\mathbf{o}} f \tag{82}$$

where $\mathbf{Y}$ is a vector with components y_{ij} (subscript i stands for the class, while j represents the subclass). This proves Proposition 1. Moreover, note that, for $p = i$ and $p \neq s$ in (80), we get

$$\frac{\partial}{\partial o_{st}}\left(\frac{y_{pj}}{y_p}\right) = 0 \tag{83}$$

which proves Proposition 2. Proposition 3 results from taking $p = s = i$, which leads to

$$\frac{\partial}{\partial o_{pt}}\left(\frac{y_{pj}}{y_p}\right) = \frac{\partial}{\partial o_{pj}}\left(\frac{y_{pt}}{y_p}\right). \tag{84}$$

Now we prove Propositions 4 and 5. We can write

$$\frac{\partial C(\mathbf{y},\mathbf{d})}{\partial o_{ij}} = \sum_{k=1}^{L} \frac{\partial C(\mathbf{y},\mathbf{d})}{\partial y_k}\frac{\partial y_k}{\partial o_{ij}}. \tag{85}$$

From Theorem 1 we know that if $C(\mathbf{y},\mathbf{d})$ is a SSB cost function, it can be written as

$$C(\mathbf{y},\mathbf{d}) = h(\mathbf{y}) + \sum_{i=1}^{L} \frac{\partial h}{\partial y_i}(d_i - y_i). \tag{86}$$

Using (86) to get the first factor of (85)

$$\frac{\partial C(\mathbf{y},\mathbf{d})}{\partial y_k} = \sum_{s=1}^{L} \frac{\partial^2 h}{\partial y_k \partial y_s}(d_s - y_s). \tag{87}$$

As $y_k = \sum_{t=1}^{n_k} y_{kt}$, the second factor of (85) can be written as follows:

$$\frac{\partial y_k}{\partial o_{ij}} = \sum_{t=1}^{n_k} \frac{\partial y_{kt}}{\partial o_{ij}}. \tag{88}$$

Hence, substituting (73), (87), and (88) in (85)

$$\frac{y_{ij}}{y_i}(y_i - d_i) = \sum_{k=1}^{L} \left(\sum_{s=1}^{L} \frac{\partial^2 h}{\partial y_k \partial y_s}(d_s - y_s) \right) \left(\sum_{t=1}^{n_k} \frac{\partial y_{kt}}{\partial o_{ij}} \right). \tag{89}$$

Using (82) we have that

$$\frac{\partial y_{kt}}{\partial o_{ij}} = \frac{\partial^2 f}{\partial o_{ij} \partial o_{kt}} \tag{90}$$

and so, we can write (89) as follows:

$$\begin{aligned} \frac{y_{ij}}{y_i}(y_i - d_i) &= \sum_{k=1}^{L} \sum_{s=1}^{L} \frac{\partial^2 h}{\partial y_k \partial y_s}(d_s - y_s) \sum_{t=1}^{n_k} \frac{\partial^2 f}{\partial o_{ij} \partial o_{kt}} \\ &= \sum_{s=1}^{L} (d_s - y_s) \sum_{k=1}^{L} \frac{\partial^2 h}{\partial y_k \partial y_s} \sum_{t=1}^{n_k} \frac{\partial^2 f}{\partial o_{ij} \partial o_{kt}}. \end{aligned} \tag{91}$$

Defining matrix $\mathbf{P_F} = \mathbf{UF}$, with size $L \times N$, (91) can be expressed in matrix form as

$$(\mathbf{d} - \mathbf{y})^T (\mathbf{H}\mathbf{P_F} + \mathbf{J}) = \mathbf{0} \tag{92}$$

where $\mathbf{H}$ is the Hessian matrix of $h(\mathbf{y})$ and matrix $\mathbf{J}$ has size of $L \times N$. Given that (92) must hold for any target vector $\mathbf{d}$, we can write the following equations:

$$\begin{aligned} &a_1(\mathbf{u}_1 - \mathbf{y})^T(\mathbf{H}\mathbf{P_F} + \mathbf{J}) = \mathbf{0} \\ &a_2(\mathbf{u}_2 - \mathbf{y})^T(\mathbf{H}\mathbf{P_F} + \mathbf{J}) = \mathbf{0} \\ &\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ &a_L(\mathbf{u}_L - \mathbf{y})^T(\mathbf{H}\mathbf{P_F} + \mathbf{J}) = \mathbf{0} \end{aligned}$$

for any real numbers $a_i, i = 1, \ldots, L$. If we add these equations, we have

$$\sum_{i=1}^{L} a_i(\mathbf{u}_i - \mathbf{y})^T(\mathbf{H}\mathbf{P_F} + \mathbf{J}) = \mathbf{0}. \tag{93}$$

Taking a_i such as $\sum_{i=1}^{L} a_i = 0$, it is easy to see that (93) can be written as

$$\mathbf{a}^T(\mathbf{H}\mathbf{P_F} + \mathbf{J}) = \mathbf{0} \tag{94}$$

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

1) Note that the sum of the elements for any column vector in matrix $\mathbf{P_F}$ is in $\mathcal{A}_L$,

$$\begin{aligned} \sum_{i=1}^{L} \mathbf{P_F}(i,j) &= \sum_{i=1}^{L} \sum_{l=1}^{n_i} \frac{\partial^2 f}{\partial o_{il} \partial o_{st}} \\ &= \frac{\partial}{\partial o_{st}} \left(\sum_{i=1}^{L} \sum_{l=1}^{n_i} \frac{\partial f}{\partial o_{il}} \right) \\ &= \frac{\partial}{\partial o_{st}} \left(\sum_{i=1}^{L} y_i \right) = 0. \end{aligned} \tag{95}$$

Therefore

$$\mathbf{P_F^T}\mathbf{H}\mathbf{P_F} + \mathbf{P_F^T}\mathbf{J} = \mathbf{0}. \tag{96}$$

That is

$$\mathbf{P_F^T}\mathbf{H}\mathbf{P_F} = -\mathbf{P_F^T}\mathbf{J}. \tag{97}$$

Multiplying both sides by a vector $\mathbf{r}_g \in \mathbb{R}^N$, we have

$$\left(\mathbf{r}_g^T \mathbf{P_F^T}\right) \mathbf{H}(\mathbf{P_F}\mathbf{r_g}) = -\mathbf{r_g^T}\mathbf{P_F^T}\mathbf{J}\mathbf{r_g}. \tag{98}$$

Since h is strictly convex in $\mathcal{P}_L$, $\mathbf{H}$ is definite negative in $\mathcal{A}_L$ and we can write

$$\mathbf{k}^T\mathbf{H}\mathbf{k} < 0, \quad \forall \mathbf{k} \neq \mathbf{0}. \tag{99}$$

Comparing the first part of (98) with (99) and taking into account that $\mathbf{P_F}\mathbf{r_g} \in \mathbb{R}^L$, we can state that

$$\mathbf{r_g^T}\mathbf{P_F^T}\mathbf{J}\mathbf{r_g} > \mathbf{0} \Leftrightarrow \mathbf{P_F}\mathbf{r_g} \neq \mathbf{0}. \tag{100}$$

We can express (100) as

$$\mathbf{r_g^T}\mathbf{P_F^T}\mathbf{J}\mathbf{r_g} \geq \mathbf{0}, \quad \forall \mathbf{u_g} \in \mathbb{R}^N \tag{101}$$

showing that matrix $\mathbf{X} = \mathbf{P_F^T}\mathbf{J}$ is semidefinite–positive in $\mathbb{R}^N$.

2) Defining matrix $\mathbf{B} = \mathbf{H}\mathbf{P_F} + \mathbf{J}$, from (94), we know that $\mathbf{a}^T\mathbf{B} = \mathbf{0}$ for any vector $\mathbf{a} \in \mathcal{A}_L$. As $\dim(\mathcal{A}_L) = L - 1$, we have that $\text{rank}(\mathbf{B}) \leq 1$. Moreover, since

$$\mathbf{J} = \mathbf{B} - \mathbf{H}\mathbf{P_F} \tag{102}$$

and $\text{rank}(\mathbf{J}) = L$, we get

$$L \leq \text{rank}(\mathbf{B}) + \text{rank}(\mathbf{H}\mathbf{P_F}) \leq 1 + \text{rank}(\mathbf{H}\mathbf{P_F}) \tag{103}$$

$$L - 1 \leq \text{rank}(\mathbf{H}\mathbf{P_F}) \leq \text{rank}(\mathbf{P_F}). \tag{104}$$

Besides, since all $\mathbf{P_F}$ columns are in $\mathcal{A}_L$, $\text{rank}(\mathbf{P_F}) < L$, and we conclude that

$$\text{rank}(\mathbf{P_F}) = L - 1. \tag{105}$$

B. Proof of Theorem 3

Let us assume that $\mathbf{Y} = \mathbf{g}(\mathbf{o})$ is the gradient of a potential function $f(\mathbf{o})$, and vector $\mathbf{z}_i = (z_{1\,|\,i}, \ldots, z_{n_i\,|\,i})$ is the gradient of a potential function f_i, that is, $\mathbf{z}_i = \nabla_{\mathbf{o}_i} f_i$.

Let us define

$$c_i(\mathbf{o}) = f(\mathbf{o}) - f_i(\mathbf{o}_i) \tag{106}$$

where $\mathbf{o}_i$ is the subvector of $\mathbf{o}$ with elements associated to class i. In the following, we demonstrate that

$$C(\mathbf{o}, \mathbf{d}) = \sum_{k=1}^{L} d_k c_k(\mathbf{o}) \tag{107}$$

is an SSB cost leading to rule

$$\mathbf{w}_{ij}(k+1) = \mathbf{w}_{ij}(k) + \rho \frac{y_{ij}}{y_i}(d_i - y_i)\mathbf{x}. \tag{108}$$

Since

$$\begin{aligned}\frac{\partial C(\mathbf{o}, \mathbf{d})}{\partial o_{ij}} &= \sum_{k=1}^{L} d_k \frac{\partial c_k(\mathbf{o})}{\partial o_{ij}} \\ &= \sum_{k=1}^{L} d_k \left(\frac{\partial f}{\partial o_{ij}} - \frac{\partial f_k(\mathbf{o}_k)}{\partial o_{ij}} \right) \\ &= \sum_{k=1}^{L} d_k \left(y_{ij} - \delta_{k-i} \frac{y_{ij}}{y_i} \right) \\ &= y_{ij} - d_i \frac{y_{ij}}{y_i} = \frac{y_{ij}}{y_i}(y_i - d_i)\end{aligned} \tag{109}$$

then learning rule (108) minimizes cost $C(\mathbf{o}, \mathbf{d})$ with a singular point when

$$\frac{\partial E\{C(\mathbf{o}, \mathbf{d}) \,|\, \mathbf{x}\}}{\partial o_{ij}} = E\left\{ \frac{\partial C(\mathbf{o}, \mathbf{d})}{\partial o_{ij}} \middle| \mathbf{x} \right\} = \frac{y_{ij}}{y_i}(y_i - p_i) = 0 \tag{110}$$

i.e., when outputs are probabilities. It remains to be proved that this point is a minimum. To do so, we calculate the second derivative. Since the only singular point corresponds to $\mathbf{y} = \mathbf{p}$, to verify that this point is a minimum, it is suffices to prove that the Hessian matrix of $E_0 = E\{C(\mathbf{o}, \mathbf{d}) \,|\, \mathbf{x}\}$ is definite or semidefinite–positive in $\mathbf{y} = \mathbf{p}$. From (110)

$$\begin{aligned}\frac{\partial^2 E_0}{\partial o_{st} \partial o_{ij}} &= \frac{\partial}{\partial o_{st}} \left(\frac{\partial E\{C(\mathbf{o}, \mathbf{d}) \,|\, \mathbf{x}\}}{\partial o_{ij}} \right) \\ &= \frac{\partial y_{ij}}{\partial o_{st}} - p_i \frac{\partial}{\partial o_{st}} \left(\frac{y_{ij}}{y_i} \right) \\ &= \frac{\partial y_{ij}}{\partial o_{st}} - p_i \left(\frac{1}{y_i} \frac{\partial y_{ij}}{\partial o_{st}} - \frac{y_{ij}}{y_i^2} \frac{\partial y_i}{\partial o_{st}} \right).\end{aligned} \tag{111}$$

Using that $\mathbf{Y} = \nabla_{\mathbf{o}} f \Rightarrow (\partial y_{ij})/(\partial o_{st}) = (\partial^2 f)/(\partial o_{st} \partial o_{ij})$, and substituting it in (111), we can write

$$\begin{aligned}\left. \frac{\partial^2 E_0}{\partial o_{st} \partial o_{ij}} \right|_{\mathbf{y}=\mathbf{p}} &= \frac{y_{ij}}{y_i} \frac{\partial y_i}{\partial o_{st}} = \frac{y_{ij}}{y_i} \frac{\partial}{\partial o_{st}} \left(\sum_{k=1}^{n} y_{ik} \right) \\ &= \frac{y_{ij}}{y_i} \sum_{k=1}^{n} \frac{\partial^2 f}{\partial o_{ik} \partial o_{st}}.\end{aligned} \tag{112}$$

Comparing (112) to (34), we conclude that

$$\mathbf{E}_0 \,|_{\mathbf{y}=\mathbf{p}} = \mathbf{X} \tag{113}$$

where $\mathbf{E}_0$ represents the Hessian matrix of E_0. As $\mathbf{X}$ is semidefinite–positive in $\mathbb{R}^N$, then $\mathbf{E_0}$ is semidefinite–positive at singular point $\mathbf{y} = \mathbf{p}$, proving in this way that $E_0 = E\{C(\mathbf{o}, \mathbf{d}) \,|\, \mathbf{x}\}$ has a minimum in $\mathbf{y} = \mathbf{p}$.

In the following, we will show that C is function of $\mathbf{Y}$. First, notice that

$$\sum_{i=1}^{L} y_i = \sum_{i=1}^{L} \left(\sum_{j=1}^{n_i} y_{ij} \right) = \sum_{i=1}^{L} \sum_{j=1}^{n_i} \frac{\partial f}{\partial o_{ij}} = (\nabla_{\mathbf{o}} f)^T \mathbf{1}_g = 1 \tag{114}$$

where $\mathbf{1}_g = (1, 1, \ldots, 1)$ is a vector of N elements. Second

$$\sum_{k=1}^{n_j} \frac{y_{jk}}{y_j} = \sum_{k=1}^{n_j} z_{k\,|\,j} = \left(\nabla_{\mathbf{o}_j} f_j \right)^T \mathbf{1}_j = 1 \tag{115}$$

where $\mathbf{1}_j = (1, 1, \ldots, 1)$ is a vector of n_j elements.

Using (114) and (115), we can deduce that the directional derivatives of f and f_j along the lines driven by $\mathbf{1}_g$ and $\mathbf{1}_j$, respectively, are constant. Therefore, we can write that

$$f(\mathbf{o} + \lambda \mathbf{1}_g) - f(\mathbf{o}) = \lambda \tag{116}$$

$$f_j(\mathbf{o}_j + \lambda \mathbf{1}_j) - f_j(\mathbf{o}_j) = \lambda \tag{117}$$

for every λ. According to this and using (106)

$$\begin{aligned}c_j(\mathbf{o} + \lambda \mathbf{1}_g) - c_j(\mathbf{o}) &= f(\mathbf{o} + \lambda \mathbf{1}_g) - f(\mathbf{o}) \\ &\quad - (f_j(\mathbf{o}_j + \lambda \mathbf{1}_j) - f_j(\mathbf{o}_j)) \\ &= \lambda - \lambda = 0.\end{aligned} \tag{118}$$

Applying the gradient operator ∇ to both sides of (116)

$$\nabla_{\mathbf{o}} f(\mathbf{o} + \lambda \mathbf{1}_g) - \nabla_{\mathbf{o}} f(\mathbf{o}) = 0 \tag{119}$$

and considering that $\mathbf{Y} = \nabla_{\mathbf{o}} f = \mathbf{g}(\mathbf{o})$, we deduce that

$$g(\mathbf{o} + \lambda \mathbf{1}_g) = g(\mathbf{o}). \tag{120}$$

From (118) and (120) it is shown that, for any vector $\mathbf{v} \in \{\lambda \mathbf{1}_g, \lambda \in \mathbb{R}\}$, the cost function (107) is, in fact, a function of vectors $\mathbf{Y}$ and $\mathbf{d}$: $C(\mathbf{Y}, \mathbf{d})$.

REFERENCES

[1] S. Amari, "Backpropagation and stochastic gradient descent method," *Neurocomput.*, vol. 5, pp. 185–196, 1993.

[2] J. I. Arribas, J. Cid-Sueiro, T. Adali, and A. R. Figueiras-Vidal, "Neural architectures for parametric estimation of *a posteriori* probabilities by constrained conditional density functions," in *Proc. Int. Conf. Neural Networks Signal Processing (NNSP)*, Aug. 1999, pp. 263–272.

[3] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Oxford Univ. Press, 1995.

[4] J. Cid-Sueiro, J. I. Arribas, S. Urbán-Muñoz, and A. R. Figueiras-Vidal, "Cost functions to estimate 'a posteriori' probabilities in multiclass problems," *IEEE Trans. Neural Netw.*, vol. 10, no. 3, pp. 645–656, May 1999.

[5] J. Cid-Sueiro and A. R. Figueiras-Vidal, "On the structure of strict sense bayesian cost functions and its applications," *IEEE Trans. Neural Netw.*, vol. 12, no. 3, May 2001.

[6] L. Devroye and L. Györfi, *Nonparametric Density Estimation: The L_1 View*. New York: Wiley, 1985.

[7] L. Devroye, L. Gyrfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*. New York: Springer-Verlag, 1996.

[8] W. Duch and N. Jankowski, "Survey of neural transfer functions," *Neural Comput. Surv.*, no. 2, pp. 163–213, 1999.

[9] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.
[10] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. New York: Academic, 1990.
[11] H. P. Graf and L. D. Jackel, "Analog electronic neural networks circuits," *IEEE Circuits Devices Mag.*, no. 55, pp. 44–49, 1989.
[12] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Englewood Cliffs, NJ: Prentice-Hall, 1995.
[13] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computat.*, vol. 3, no. 1, pp. 79–87, 1991.
[14] J. W. Miller, R. Goodman, and P. Smyth, "On loss functions which minimize to conditional expected values and posterior probabilities," *IEEE Trans. Inf. Theory*, vol. 39, no. 4, pp. 1404–1408, Jul. 1993.
[15] E. A. Nadaraya, "On estimating regression," *Theor. Probab. Appl.*, vol. 9, pp. 141–142, 1964.
[16] M. Saerens, P. Latinne, and C. Dekaestecker, "Any reasonable cost function can be used for *a posteriori* probability approximation," *IEEE Trans. Neural Netw.*, vol. 13, no. 5, Sep. 2002.
[17] D. F. Specht, "Probabilistic neural networks," *Neural Netw.*, vol. 3, pp. 109–118, 1990.
[18] ——, "A general regression neural network," *IEEE Trans. Neural Netw.*, vol. 2, no. 6, pp. 568–576, Nov. 1991.
[19] B. A. Telfer and H. H. Szu, "Energy functions for minimizing misclassification error with minimum-complexity networks," *Neural Netw.*, vol. 7, no. 5, pp. 809–918, 1994.
[20] G. S. Watson, "Smooth regression analysis," *Sankhya—Indian J. Statist.*, ser. A, vol. 26, pp. 359–372, 1964.
[21] B. S. Wittner and J. S. Denker, "Strategies for teaching layered neural networks classification tasks," in *Neural Information Processing Systems*, D. Z. Anderson, Ed. Denver, CO: Amer. Inst. Phys., 1988, pp. 850–859.
[22] J. M. Zurada, *Introduction to Artificial Neural Systems*. St. Paul, MN: West, 1992.

Inma Mora-Jiménez (M'04) received the degree in telecommunications engineering from the University Politécnica de Valencia, Valencia, Spain, in 1998, and the Ph.D. degree from the University Carlos III de Madrid, Leganés-Madrid, Spain, in 2004.

She is currently a Teaching Assistant in the Department of Signal Theory and Communications, University Carlos III de Madrid. Her main research topics include machine learning and applied fields such as data mining and digital image processing.

Jesús Cid-Sueiro (M'95) received the Telecomm Engineer degree from the Universidad de Vigo, Vigo, Spain, in 1990 and the Ph.D. degree from Universidad Politecnica Madrid, Madrid, Spain, in 1994.

Since 1999, he has been an Associate Professor in the Department of Signal Theory and Communications, Universidad Carlos III de Madrid, Madrid, Spain. His main research interests include statistical learning theory, neural networks, Bayesian methods and their applications to communications, multimedia signal processing, and education.