Neural Networks 21 (2008) 1302–1310

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Data classification with multilayer perceptrons using a generalized error function

Luís M. Silva [a,*], J. Marques de Sá [a,b], Luís A. Alexandre [c,d]

[a] INEB-Instituto de Engenharia Biomédica, Porto, Portugal
[b] Faculdade de Engenharia, Universidade do Porto, Porto, Portugal
[c] Dep. de Informática, Universidade da Beira Interior, Covilhã, Portugal
[d] IT, Networks and Multimedia Group, Covilhã, Portugal

ARTICLE INFO

Article history:
Received 9 November 2007
Received in revised form
31 March 2008
Accepted 28 April 2008

Keywords:
Multilayer perceptrons
Data classification
Error functions

ABSTRACT

The learning process of a multilayer perceptron requires the optimization of an error function $E(\mathbf{y}, \mathbf{t})$ comparing the predicted output, $\mathbf{y}$, and the observed target, $\mathbf{t}$. We review some usual error functions, analyze their mathematical properties for data classification purposes, and introduce a new one, E_{Exp}, inspired by the Z-EDM algorithm that we have recently proposed. An important property of E_{Exp} is its ability to emulate the behavior of other error functions by the sole adjustment of a real-valued parameter. In other words, E_{Exp} is a sort of generalized error function embodying complementary features of other functions. The experimental results show that the flexibility of the new, generalized, error function allows one to obtain the best results achievable with the other functions with a performance improvement in some cases.

1. Introduction

Multi-layer perceptrons (MLP) are a popular form of feedforward artificial neural networks with many successful applications in data classification. The supervised learning (training) process of an MLP with input data $\mathbf{x}$ and target $\mathbf{t}$, requires the use of an objective function (or error/cost/loss function) $E(\mathbf{y}, \mathbf{t})$ in order to assess the deviation of the predicted output values, $\mathbf{y} = \text{MLP}(\mathbf{x}; \mathbf{w})$ from the observed data values $\mathbf{t}$ and use that assessment for the convergence towards an optimal set of weights $\mathbf{w}^*$. There are many MLP training algorithms using the $\frac{\partial E}{\partial w}$ gradient information either directly or indirectly. In the present paper we concentrate on using the well-known backpropagation (BP) algorithm without loss of generalization of the main conclusions of the paper.

In what concerns the error function $E(\mathbf{y}, \mathbf{t})$, the well-known mean square error (MSE) function is by far the most commonly used, but as we will discuss later, it is not the most appropriate for data classification problems. There are other alternatives, such as the cross entropy (CE) error function and other entropy-based functions, which have been specifically applied by us to data classification problems (Santos, Alexandre, & Marques de Sá, 2004; Silva, Marques de Sá, & Alexandre, 2005). We have also proposed a new error function inspired on entropic criteria: the error density at the origin (Z-EDM) (Silva, Alexandre, & Marques de Sá, 2005, 2006). In this paper, we review these error functions, analyze their mathematical properties for data classification purposes and clarify aspects that are commonly omitted or have been misinterpreted in the literature. We also propose a new error function, E_{Exp}, inspired by the Z-EDM, which is capable of emulating the behavior of other error functions by the adjustment of a single real-valued parameter. Experimental results show that the flexibility of this new, generalized, exponential error function, E_{Exp}, allows one to achieve the best results achievable with the other functions, capitalizing on their complementary properties, with a performance improvement in some situations.

The paper is organized as follows: sections two to four present some error functions and study their behaviors in terms of the corresponding gradient; section five deals with monotonic error functions and presents the new error function; experimental results are reported in section six and finally the paper ends with some concluding remarks.

2. Usual error functions

We consider the usual classification problem where a pattern $\mathbf{x} \in \mathbb{R}^d$ is to be assigned to one of C classes, $(\omega_1, \ldots, \omega_C)$, by an MLP with one hidden layer and weight vector $\mathbf{w}$. The MLP is trained using a set of training pairs $(\mathbf{x}_i, \mathbf{t}_i)$, $i = 1, \ldots, N$, where each $\mathbf{t}_i = (t_{1,i}, \ldots, t_{C,i})$ is a realization of a (target) variable $\mathbf{t} = (t_1, \ldots, t_C)$ that describes the class to which $\mathbf{x}_i$ belongs in an *1-out-of-C* coding, with $t_k \in \{0, 1\}$ or $t_k \in \{-1, 1\}$ for $k = 1, \ldots, C$. Hence, the MLP

* Corresponding address: INEB - Instituto de Engenharia Biomedica, Campus da FEUPRua Dr. Roberto Frias, s/n 4200-465 Porto, Portugal. Tel.: +351 22 508 16 23; fax: +351 22 508 16 24.
E-mail addresses: lmsilva@fe.up.pt (L.M. Silva), jmsa@fe.up.pt (J. Marques de Sá), lfbaa@di.ubi.pt (L.A. Alexandre).

0893-6080/$ – see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2008.04.004

has an output layer described by a vector $\mathbf{y} = (y_1, \ldots, y_C)$ that produces for each $\mathbf{x}_i$ its corresponding output $\mathbf{y}_i = (y_{1,i}, \ldots, y_{C,i})$. We assume this setting throughout the rest of the paper.

2.1. Mean square error

The MSE function is expressed as

$$E_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{t}_i - \mathbf{y}_i\|^2. \tag{1}$$

Originally derived for regression problems, the MSE function can be obtained by the maximum likelihood (ML) principle assuming the independence and Gaussianity of the target data (see Bishop (1995) for a detailed derivation of E_{MSE}).

Note, however, that the Gaussianity assumption of the target data in classification is not valid, due to its discrete nature (representing discrete class labels). Nevertheless, it can be shown (see below) that when using an *1-out-of-C* coding scheme for the targets, the MSE trained outputs of the network approximate the posterior probabilities of the class membership, $y_k = \hat{P}(\omega_k|\mathbf{x})$.

2.2. Cross-entropy

The CE cost function can also be derived from the maximum likelihood principle. Each component $y_k, k = 1, \ldots, C$ of the output vector is interpreted as an estimate of the posterior probability that input pattern $\mathbf{x}$ belongs to class ω_k, $y_k = \hat{P}(\omega_k|\mathbf{x})$ associated with a "true" distribution $\mathbf{p} = (p_1, \ldots, p_C)$ where $p_k = P(\omega_k|\mathbf{x}),\ k = 1, \ldots, C$.

Assuming that the classes are mutually exclusive, the true, $p(\mathbf{t}|\mathbf{x})$, and neural network, $p_{\mathbf{w}}(\mathbf{t}|\mathbf{x})$, probabilistic models for $\mathbf{t}$ can be described by the multinomial distributions:

$$p(\mathbf{t}|\mathbf{x}) = p_1^{t_1} p_2^{t_2} \ldots p_C^{t_C} \tag{2}$$

$$p_{\mathbf{w}}(\mathbf{t}|\mathbf{x}) = y_1^{t_1} y_2^{t_2} \ldots y_C^{t_C}. \tag{3}$$

We would like the model in (3) to approximate the true distribution (2). For that purpose we may apply the ML principle, or, equivalently, attempt to minimize the following Kulback-Leibler divergence which measures how well (3) approximates (2)

$$\sum_{i=1}^{N} \log\left(\frac{p(\mathbf{t}_i|\mathbf{x}_i)}{p_{\mathbf{w}}(\mathbf{t}_i|\mathbf{x}_i)}\right) = \sum_{i=1}^{N} \log\left(\frac{p_{1,i}^{t_{1,i}} \ldots p_{C,i}^{t_{C,i}}}{y_{1,i}^{t_{1,i}} \ldots y_{C,i}^{t_{C,i}}}\right)$$

$$= -\sum_{i=1}^{N}\sum_{k=1}^{C} t_{k,i} \log\left(y_{k,i}\right) + \sum_{i=1}^{N}\sum_{k=1}^{C} t_{k,i} \log\left(p_{k,i}\right). \tag{4}$$

Note that, as the values $p_{k,i} = P(\omega_k|\mathbf{x}_i)$ are unknown, (4) cannot be used as an error function. However, the $p_{k,i}$ do not depend on the parameters $\mathbf{w}$ of the MLP, which means that the minimization of (4) is equivalent to the minimization of

$$E_{CE} = -\sum_{i=1}^{N}\sum_{k=1}^{C} t_{k,i} \log\left(y_{k,i}\right). \tag{5}$$

Expression (5) is known in the literature as the *cross-entropy* error function. For the two-class case we only need one output such that $y = \hat{P}(\omega_1|\mathbf{x})$ $(1 - y = \hat{P}(\omega_2|\mathbf{x}))$ and the Bernoulli distribution is used for $p(t|\mathbf{x})$ and $p_{\mathbf{w}}(t|\mathbf{x})$. The designation *cross-entropy* associated to expression (5) may cause some confusion. In fact, the cross-entropy between two distributions p and q, which can be used as a measure of their discrepancy is defined as

$$-\sum_{\mathbf{x}} p(\mathbf{x}) \log q(\mathbf{x}). \tag{6}$$

Its minimum value is obtained when $p = q$. Expression (5) is similar to (6) but one must note that the $t_{k,i}$, or more precisely $\mathbf{t}_i$, are *not* probabilities (are in fact random vectors with multinomial distribution). The role of $p(\mathbf{x})$ in (6) is played by the unknown $p(\mathbf{t}|\mathbf{x})$ as defined above. Since it is not dependent on the network's parameters, it is of no consequence for the minimization process, and thus, can be disregarded. Moreover, expression (6) is the distribution version of the cross-entropy, while (5) is the empirical one. There are some authors that interpret the $t_{k,i}$ as $P(\omega_k|\mathbf{x}_i)$. This is incorrect since as $t_{k,i} \in \{0, 1\}$ this would mean that a pattern would always be correctly classified. In fact, the $t_{k,i}$ in (5) are just acting as switches. When a particular $t_{k,i} = 1$ (which means that $\mathbf{x}_i$ belongs to class ω_k), then $y_{k,i}$ must be maximum and thus we just minimize $-\log(y_{k,i})$ (all the other $t_{j,i} = 0,\ j \neq k$).

2.3. Mean square error or cross entropy

From the above discussion it seems natural to choose the Kullback–Leibler or more precisely the cross-entropy error function to train neural network classifiers, because when interpreting the outputs as probabilities this is the optimal solution. In fact, the CE error function takes into account the binary characteristic of the targets. Several authors have studied the conditions that the outputs of a neural network must satisfy in order to use them as estimators of the posterior probabilities. In Gish (1990) and Richard and Lippmann (1991) it is shown that for an *1-out-of-C* coding scheme, with large N and a number of samples in each class that reflects the prior probabilities, networks trained with MSE provide outputs that are approximations of posterior probabilities. These authors also derived the CE cost function from the maximum likelihood or maximum mutual information principles and have arrived to the same conclusions. There are other reasons to choose CE. Several authors reported marked reductions on convergence rates and density of local minima (Matsuoka & Yi, 1991; Solla, Levin, & Fleisher, 1988) due to the characteristic steepness of the CE function. In fact, it is easy to see that slight changes on the output of the network have more effect when using CE than MSE, because cancelations in the error gradients generate high error gradients for outputs very distant from their targets. As a function of the absolute errors, MSE tends to produce large relative errors for small output values. As a function of the relative errors, CE is expected to estimate more accurately small probabilities (Baum & Wilczek, 1987; Bishop, 1995; Gish, 1990; Hinton, 1989; Solla et al., 1988).

3. Zero-error density maximization

Let us now define the error (deviation) variable $\mathbf{e} = \mathbf{t} - \mathbf{y}$. One can easily see that when using the *1-out-of-C* output coding, the errors regarding each class lie in disjoint hypercubes with the origin as their unique common point. The three-class case is represented in Fig. 1 assuming the following *1-out-of-C* coding: $\mathbf{t}(\omega_1) = (1, -1, -1)$, $\mathbf{t}(\omega_2) = (-1, 1, -1)$ and $\mathbf{t}(\omega_3) = (-1, -1, 1)$. Therefore, $\mathbf{e} = [-2, 2]^3$ with the errors in each class varying in a distinct hypercube. For instance, the ω_1 error corresponds to the hypercube $\mathbf{t}(\omega_1) - \mathbf{y} = (1, -1, -1) - (y_1, y_2, y_3) \in [0, 2] \times [-2, 0] \times [-2, 0]$, i.e., the rightmost shadowed hypercube in Fig. 1.

We would optimally like that during the training process of an MLP the output $\mathbf{y}$ gets as close as possible to the target $\mathbf{t}$ and thus the errors (deviations) convergent to the origin. For a given data set $\{(\mathbf{x}_i, \mathbf{t}_i)|i = 1, \ldots, N\}$, we would then get in this optimal scenario $\mathbf{e}_i = \mathbf{t}_i - \mathbf{y}_i = \mathbf{0}\ \forall i$, which amounts to a δ-Dirac distribution of the error variable centered at the origin. As a matter of fact, when using entropic error functions (Santos et al., 2004; Silva, Marques de Sá et al., 2005) we observe the tendency as training evolves of reaching higher peak distribution of the errors at the origin (the δ-

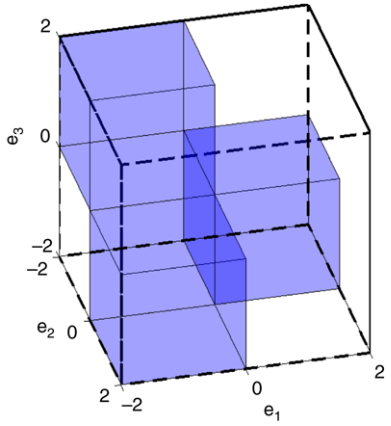


Fig. 1. Support space (shadowed cubes) for the error distribution in a three-class ($C = 3$) problem, $\mathbf{e} = (e_1, e_2, e_3)$.

Dirac distribution is the one with minimum entropy). This idea led us to adjust the weight vector $\mathbf{w}$ by maximizing the error density at the origin

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmax}} f(\mathbf{0}; \mathbf{w}), \tag{7}$$

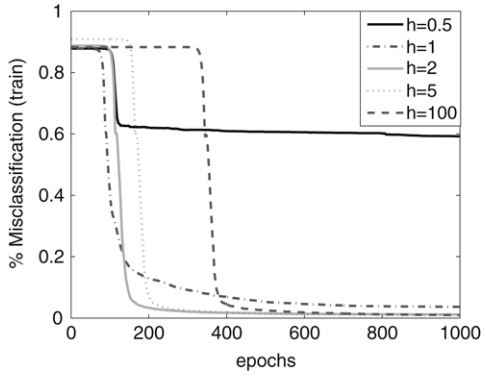
where $\mathbf{w}^*$ is the sought for optimal weight vector for the MLP and f is the error density (parameterized by $\mathbf{w}$). In practice, the error distribution is not known and making parametric assumptions would be very restrictive. Thus, we rely on nonparametric density estimation by using the well-known kernel density estimation procedure of Parzen windows (Silverman, 1986). Given a set of errors $\mathbf{e}_1, \ldots, \mathbf{e}_N$, the estimated density at $\mathbf{e} = \mathbf{0}$ is

$$\hat{f}(\mathbf{0}) = \frac{1}{Nh^C} \sum_{i=1}^{N} K\left(\frac{\mathbf{0} - \mathbf{e}_i}{h}\right), \tag{8}$$

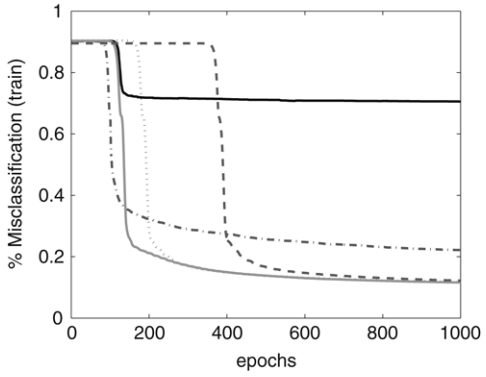
where K is a multidimensional kernel function, h is the smoothing parameter (kernel bandwidth) and C the dimension of $\mathbf{e}$ (number of classes). The multivariate Gaussian kernel with zero mean and unit covariance (Silverman, 1986) is chosen for K. This is because of its continuity and differentiability, which are crucial properties for the BP algorithm. Also, as proven in Silva et al. (2006), the Gaussian kernel satisfies the conditions needed to ensuring that the use of kernel density estimation does not affect the optimal solution. The final expression to be maximized becomes:

$$\hat{f}(\mathbf{0}) = \frac{1}{Nh^C} \sum_{i=1}^{N} \frac{1}{(2\pi)^{C/2}} \exp\left(-\frac{\mathbf{e}_i^2}{2h^2}\right). \tag{9}$$

This procedure, named Zero-Error Density Maximization (Z-EDM), inspired by our previous work on using entropic error measures, can be easily plugged in the usual backpropagation scheme as described in Silva, Alexandre et al. (2005) and Silva et al. (2006). Note that expression (9) depends on the kernel bandwidth h. This parameter controls the smoothness of the density estimate and consequently the smoothness of the error function. In order to better understand the influence of h in the training process several data sets were used for training 100 times (full data set) using several different values of h. Fig. 2 shows the mean training curves (the standard deviations are small) for two data sets: Olive and Ctg16. We found that a value of h smaller than 1 does not work, independently of the data set, number of classes and/or the number of training examples. When h is increased, the curve is basically shifted forward and a flat region appears in the earlier epochs. This general behavior was found to be the same for all tested data sets, regardless of the number of classes and number of examples available for training.



(a) Olive: 9 classes, 8 features and 572 patterns.



(b) Ctg16: 10 classes, 16 features and 2126 patterns.

Fig. 2. Mean training curves with Z-EDM for different values of h in two data sets.

Let us now consider, for simplicity's sake, an MLP with one output for a two-class problem. The gradient of (9) with respect to a particular parameter w (weight of the MLP) is derived as

$$\frac{\partial \hat{f}(0)}{\partial w} = -\frac{1}{Nh} \sum_{i=1}^{N} \frac{e_i}{h^2\sqrt{2\pi}} \left[\exp\left(-\frac{e_i^2}{2h^2}\right)\right] \frac{\partial e_i}{\partial w}. \tag{10}$$

In this expression we may consider the function

$$\varphi(e) = \frac{e}{Nh^3\sqrt{2\pi}} \exp\left(-\frac{e^2}{2h^2}\right) \quad e \in [-2, 2] \tag{11}$$

as a weight function of the gradient "particle" $\frac{\partial e}{\partial w}$. Fig. 3 shows $\varphi(e)$ for some values of N and h. We can see that gradient particles corresponding to larger absolute values of e get larger weights, whereas gradient particles corresponding to smaller values of e will have a small contribution to the update value (10) of the parameter w. Of course, if we increase N or h, then $\varphi(e) \to 0$ and this is the reason for the initial flat platforms encountered in the first epochs of the training error. If we also look to the order of magnitude of the values given by (11), we can conclude that this behavior is due to the initial convergence "effort" being done by the adaptive learning rate procedure[1] while attempting to compensate those minuscule orders of magnitude.

In fact, we can make some modifications to our error function in order to avoid this problem. Note that the maximization of (9) is equivalent to the maximization of

$$E_{ZEDM} = \sum_{i=1}^{N} h^2 \exp\left(-\frac{\mathbf{e}_i^2}{2h^2}\right) \tag{12}$$

[1] We use an adaptive learning rate procedure as described in Silva, Alexandre et al. (2005).

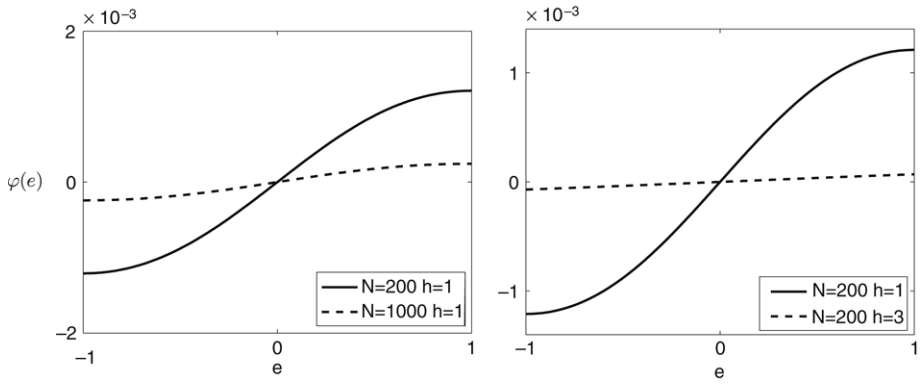


Fig. 3. $\varphi(e)$ as in (11) for different values of N and h. For better visualization we restricted e to the interval $[-1,\ 1]$.

in the sense that the same solutions are encountered, because $\frac{1}{Nh^C(2\pi)^{C/2}}$ and h^2 are just positive scaling factors (we use the factor h^2 to allow a simplification of the gradient).

4. Gradient analysis

Let us now compare the gradients of MSE, CE and Z-EDM. For simplicity we analyze the one output case (two-class problem). We have

$$\frac{\partial E_{\text{MSE}}}{\partial w} = -\sum_{i=1}^{N} e_i \frac{\partial y_i}{\partial w}, \tag{13}$$

$$\frac{\partial E_{\text{CE}}}{\partial w} = -\sum_{i=1}^{N} \frac{e_i}{y_i(1-y_i)} \frac{\partial y_i}{\partial w}, \tag{14}$$

$$\frac{\partial E_{\text{ZEDM}}}{\partial w} = \sum_{i=1}^{N} \exp\left(-\frac{e_i^2}{2h^2}\right) e_i \frac{\partial y_i}{\partial w}. \tag{15}$$

Note that we now write the gradients in terms of the gradient particle $\frac{\partial y}{\partial w}$. Consider the following weight functions

$$\varphi_{\text{MSE}}(e) = e, \tag{16}$$

$$\varphi_{\text{CE}}(y) = \frac{e}{y(1-y)} = \frac{t-y}{y(1-y)} \qquad t \in \{0, 1\}, \tag{17}$$

$$\varphi_{\text{ZEDM}}(e) = \exp\left(-\frac{e^2}{2h^2}\right) e. \tag{18}$$

Fig. 4 shows these functions for some values of their corresponding parameters. From this figure we can see the linear behavior of φ_{MSE}: the gradient particles $\frac{\partial y_i}{\partial w}$ have a weight equal to the corresponding error. The φ_{CE} function also confers larger weights to gradient particles corresponding to larger errors. Note that, when $t = 1$ ($t = 0$) larger errors correspond to y closest to zero (one). However, the weight attribution is not linear but hyperbolic. For $\varphi_{\text{Z-EDM}}$ we can distinguish three basic behaviors. When we let $h \to 0$ then $\varphi_{\text{Z-EDM}} \to 0$. If the parameters of the network are initialized in $[-b, b]$ with b close to zero, then in the early phase of the training process, all the errors are around the values -1 and 1. Thus, with h too small, the algorithm will have difficulties to converge (or even will not be able to start at all!) because $\varphi_{\text{Z-EDM}}$ gives weights close to zero. For moderate h ($h \approx 2$), $\varphi_{\text{Z-EDM}}$ is a nonlinear function (similar to a sigmoid) where, again, gradient particles corresponding to larger errors get larger weights. Note, however, the contrast with φ_{CE}: for larger errors φ_{CE} "accelerates" the weight value while $\varphi_{\text{Z-EDM}}$ "decelerates". Finally, when h is large, $\varphi_{\text{Z-EDM}}$ behaves like φ_{MSE}. In fact, it is easy to see that $\lim_{h\to+\infty} \varphi_{\text{Z-EDM}} = \varphi_{\text{MSE}}$.

5. Monotonic error functions

As discussed earlier, when a neural network is trained using MSE or CE minimization, its outputs approximate the posterior probabilities of class membership. Thus, in the presence of large data sets, it tends to produce optimal solutions in the Bayes sense. However, as argued in Hampshire and Waibel (1990) and Møller (1993), minimization of the error function does not necessarily imply misclassification minimization in practice (especially for small data sets or in the presence of local minima). Sub-optimal solutions may occur due to flat regions in weight space. This can be seen with a simple example. Let us assume a two class problem with one output *per* class. The squared error for a particular pattern $\mathbf{x}$ from class ω_1 can be written as (considering $\mathbf{t}(\omega_1) = (1, 0)$)

$$E(\mathbf{x}) = (t_1 - y_1)^2 + (t_2 - y_2)^2 \tag{19}$$

$$= (1 - y_1)^2 + y_2^2. \tag{20}$$

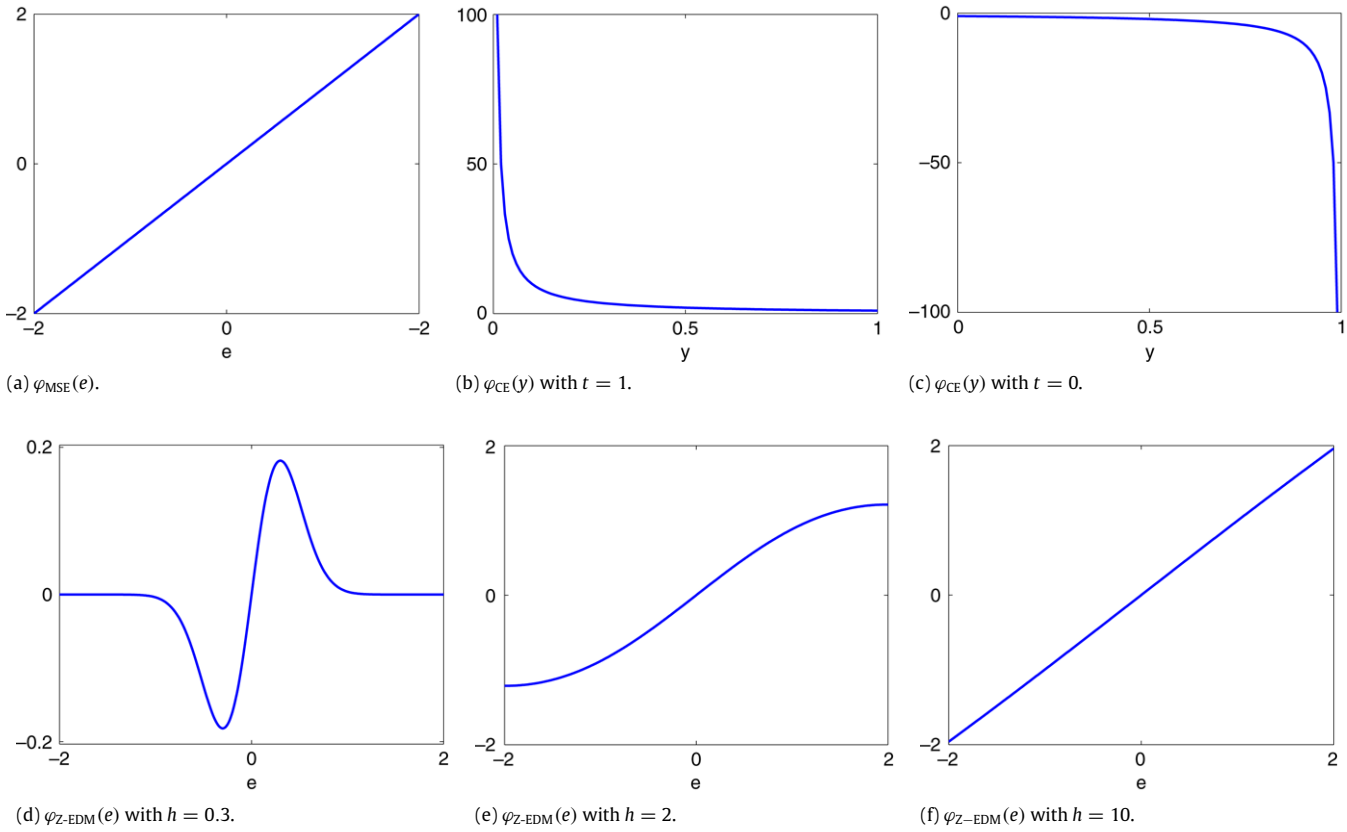
The contours of E are shown in Fig. 5. Using the rule that $\mathbf{x}$ belongs to class ω_1 if $y_1(\mathbf{x}) > y_2(\mathbf{x})$ we can see that while $\mathbf{x}_2$ is correctly classified, $\mathbf{x}_1$ is misclassified. However, $\mathbf{x}_1$ has a *lower* MSE than $\mathbf{x}_2$, which reinforces the idea that sub-optimal solutions may occur. The same happens with CE. Thus, minimization of these error functions does not imply misclassification minimization. For this reason, they were designated *non-monotonic* in Hampshire and Waibel (1990). In the same work, a monotonic error function is proposed: classification figures of merit (CFM_{mono}). This error function focuses mostly on the reduction of misclassification (this is known as *differential learning*) and not on achieving an exact convergence to the target values. However, training with CFM_{mono} was found to be much slower than with MSE or CE.

5.1. A simple monotonic error function

We can define a simple monotonic error function for a two-class problem. We consider an MLP with one output *per* class $\mathbf{y} = (y_1, y_2)$ and a class encoding defined by $\mathbf{t} = (t_1, t_2) = (1, -1)$ and $\mathbf{t} = (t_1, t_2) = (-1, 1)$ for classes ω_1 and ω_2, respectively. By the definition presented above, a monotonic error function should have contours parallel to $y_1 = y_2$. This can be achieved with the following error function:

$$E_{\text{SMF}} = \frac{1}{2} \sum_{i=1}^{N} [(y_{1i} - y_{2i}) - (t_{1i} - t_{2i})]^2. \tag{21}$$

This is a simple transformation (rotation and shifting) of the parabolic cylinder $z = x^2$. Note that as $E_{\text{SMF}} \geq 0$ it has a global minimum of $E = 0$ when the outputs equal the targets. Fig. 6 shows the error surface and corresponding contour plot for patterns from ω_1 and ω_2.



(a) $\varphi_{\text{MSE}}(e)$. (b) $\varphi_{\text{CE}}(y)$ with $t = 1$. (c) $\varphi_{\text{CE}}(y)$ with $t = 0$.

(d) $\varphi_{\text{Z-EDM}}(e)$ with $h = 0.3$. (e) $\varphi_{\text{Z-EDM}}(e)$ with $h = 2$. (f) $\varphi_{\text{Z-EDM}}(e)$ with $h = 10$.

Fig. 4. Gradient-weighting functions for MSE, CE and Z-EDM.

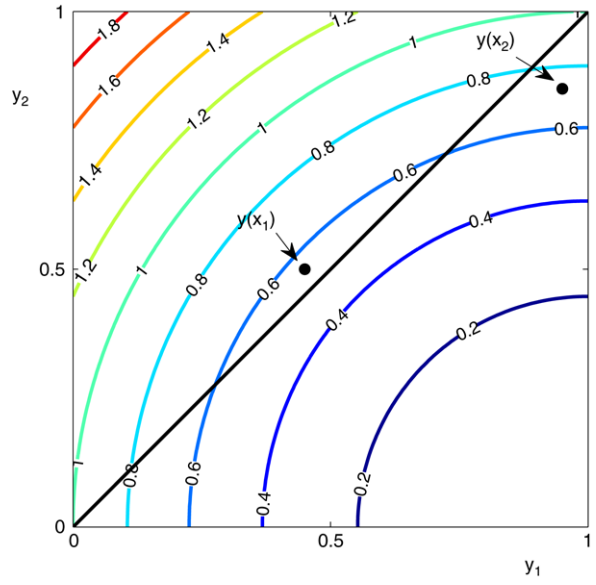


Fig. 5. Contours of MSE for a ω_1 pattern.

The gradient can be calculated as

$$\left(\frac{\partial E_{\text{SMF}}}{\partial y_{1i}}, \frac{\partial E_{\text{SMF}}}{\partial y_{2i}}\right) = ((y_{1i} - y_{2i}) - (t_{1i} - t_{2i}), \\ -[(y_{1i} - y_{2i}) - (t_{1i} - t_{2i})]), \quad (22)$$

where we note that $\frac{\partial E}{\partial y_{1i}} = -\frac{\partial E}{\partial y_{2i}}$.

The flexibility of E could be enhanced using the following version of (21):

$$E_{\text{SMF}} = \frac{1}{2}\sum_{i=1}^{N}[(y_{1i} - y_{2i}) - (t_{1i} - t_{2i})]^{\gamma}. \quad (23)$$

The parameter γ is an even integer positive number ensuring that E_{SMF} is always non-negative. The major effect of increasing γ is exemplified in Fig. 7. The steepness is increased in regions far from the desired target, whereas in regions near the desired target the function is flattened.

5.2. Exponential error function

Møller (1993) proposed an error function with a *soft-monotonicity* property, controlled by a positive parameter. It is defined as

$$E_{\text{Moller}} = \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{C}\exp\left(-\alpha(y_{k,i} - t_{k,i} + \beta)(t_{k,i} + \beta - y_{k,i})\right) \quad (24)$$

and was designated *exponential error function*. The parameters β and α are positive; β is the width of a region, $\mathcal{R}$, of acceptable error around the desired target and α controls the steepness of the error function outside that region ($\bar{\mathcal{R}}$). If we increase α then E_{Moller} becomes more steep in $\bar{\mathcal{R}}$, forcing the outputs towards the boundary of $\mathcal{R}$. By decreasing β, the outputs are pulled towards the desired targets (see Møller (1993) for a detailed discussion). When $\alpha \to +\infty$, E_{Moller} becomes monotonic, while varying α the degree of monotonicity is controled (soft-monotonicity).

5.3. Another exponential error function

The analysis of MSE, CE and Z-EDM gradient behaviors (presented in Section 4), suggested that it might be possible to

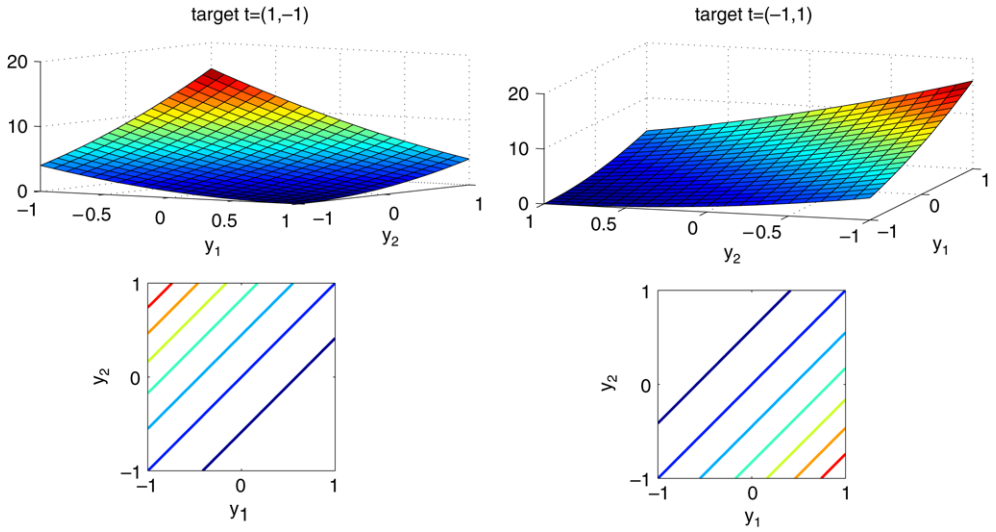


Fig. 6. *Left*: Error surface and contour plots of E_{SMF} in the presence of a pattern from ω_1; *Right*: The same but for a pattern from ω_2.

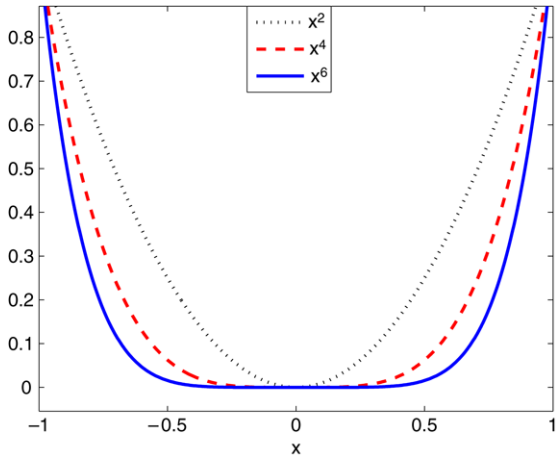


Fig. 7. Effect of increasing the power of a polynomial function x^{γ}.

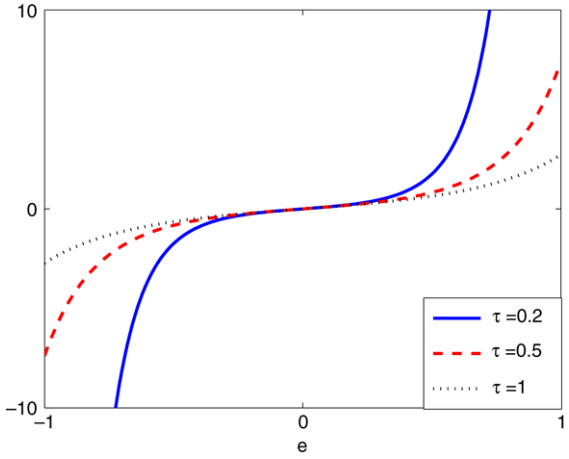


Fig. 8. Plot of the gradient of E_{Exp} for one output and different positive values of τ. For visualization purposes, e is restricted to the interval $[-1,\ 1]$.

create a parameterized error function capable of "emulating" those gradients. This error function (another exponential error function) is a generalization of (12), that allows positive arguments in the exponential function. It is expressed as

$$E_{\text{Exp}} = \sum_{i=1}^{N} \tau \exp\left(\mathbf{e}_i^2/\tau\right) = \sum_{i=1}^{N} \tau \exp\left(\frac{1}{\tau}\sum_{k=1}^{C} e_{k,i}^2\right), \tag{25}$$

where τ is a real number. Clearly, E_{Exp} resembles E_{Moller} for $\beta = 0$. There is, however, a significant difference: the location of the sum over k. In detail, in E_{Moller} we sum the exponentials of the (squares of) errors while in E_{Exp} we compute the exponential of the sum of those quantities. This implies a fundamental difference in terms of the gradients (for $\beta = 0$):

$$\frac{\partial E_{\text{Exp}}}{\partial y_{k,i}} = -2\sum_{i}\left[\exp\left(\frac{1}{\tau}\sum_{k=1}^{C} e_{k,i}^2\right)\right] e_{k,i} \tag{26}$$

$$\frac{\partial E_{\text{Moller}}}{\partial y_{k,i}} = -\sum_{i} \alpha\left[\exp\left(+\alpha e_{k,i}^2\right)\right] e_{k,i}. \tag{27}$$

We see that with E_{Exp}, the backpropagated error through the output y_k uses information from *all* other outputs (present in $\exp\left(\frac{1}{\tau}\sum_{k=1}^{C} e_{k,i}^2\right)$), while E_{Moller} only uses the error associated to that particular output (present in $\exp\left(+\alpha e_{k,i}^2\right)$). It is easy to see that if $\tau < 0$, E_{Exp} recovers the (negative) Z-EDM error function. Thus, we may also say that for $\tau \to -\infty$, E_{Exp} behaves like MSE. When $\tau > 0$, E_{Exp} behaves like CE. This can be seen in Fig. 8 where the gradient of E_{Exp} (for one output) is plotted for different positive values of τ. From small to moderate values of τ, the function has a marked hyperbolic shape, in the same sense as CE: smaller errors get smaller weights with an "accelerated" increasing when the errors get larger. The parameter τ is, as α in E_{Moller}, controlling the steepness of E_{Exp}. Also for $\tau \to \infty$, E_{Exp} behaves like MSE. In summary, E_{Exp} has the appropriate *flexibility to emulate the whole range of Z-EDM - MSE - CE behaviors with a control of the degree of monotonicity*.

6. Experiments

To evaluate and compare the capabilities of the different error functions presented, we conducted an experimental procedure using several publicly available data sets (mainly from the UCI repository (Blake & Merz, 1998)). The problems were chosen so as to comprise a wide range of real world applications as well as to include several complexities: different number of features, classes and patterns. Table 1 summarizes the main characteristics of these data sets.

Table 1
Data sets used in the experiments

Procedure 1	# patterns	# features	# classes	Source	Procedure 2	# patterns	# features	# classes	Source
Ionosphere	351	34	2	UCI	Pima	768	8	2	UCI
Liver	345	6	2	UCI	Spambase	4601	57	2	UCI
Wdbc	569	30	2	UCI	Vehicle	846	18	4	UCI
Wine	178	13	3	UCI	Vowelc	990	10	11	UCI
New Thyroid	215	5	3	UCI	Pb12	608	2	4	Jacobs et al. (1991)
Olive	572	8	9	Forina and Armanino (1981)	Ctg16	2126	16	10	Marques de Sá (2001)

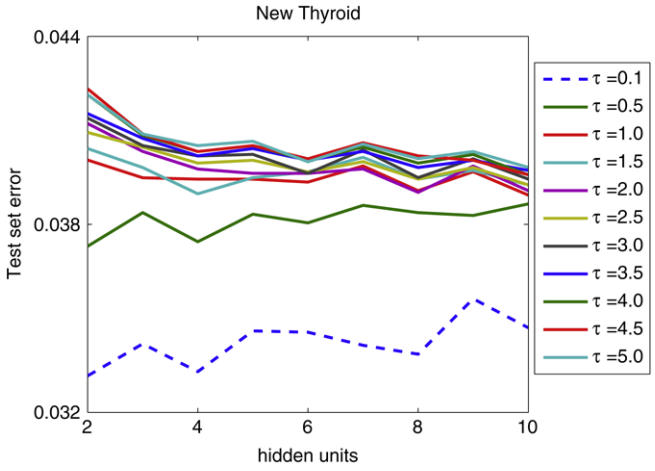


Fig. 9. Choice of τ in E_{Exp} for New Thyroid.

6.1. Procedure 1

Data sets with a small number of patterns (less than 600) were evaluated with this procedure. Each data set was randomly divided in training (50%) and test set (50%). The training set was used to train the MLP during l epochs and the test set error was recorded for each of the l epochs. The training and test sets interchange their roles and the train and test procedure is again performed. This is repeated 100 times, using different initial weights and randomized training and test sets. The minimum mean test error over the 100 repetitions is then reported. The number of hidden units in the MLP architecture is varied from 2 to 20. For a fair comparison, the initial weights and train/test set partitions of the 100 repetitions were equal for the different error functions.

Choosing the value of τ, α and γ

To use E_{Exp}, E_{Moller} and E_{SMF} one must set values for τ, α and γ, respectively. In what concerns E_{Exp} and E_{Moller}, the experiments were repeated for several values of $|\tau|$ and α, ranging from 0.1 to 10 (usually by steps of 0.5). Figures like Fig. 9 were produced to help in the choice of the best value. Some data sets "prefer" smaller values of those parameters, while others "prefer" higher ones, with an obvious and expected reversed behavior between E_{Exp} and E_{Moller}. However, there is no evident pattern for this behavior.

The E_{Exp} experiments were performed with positive and negative values (denoted τ^+ and τ^-). Note that the E_{Exp} results with τ^- corresponds to applying the Z-EDM algorithm. We found that, in general, the best choices for τ^- were low values (usually $\tau^- = -10$), with the exception of Wdbc, where $\tau^- = -1.2$. This was also the best result (with 2 hidden units) among all the error functions, a result that was already found and reported in Silva et al. (2006). These findings show that the flexibility of E_{Exp} provides a valuable option to the usual error functions. We also used $\tau^+ = 100$ and $\tau^- = -100$ to verify whether E_{Exp} behaves like MSE. These results are denoted Exp $\leftrightarrow$ MSE(+) and Exp $\leftrightarrow$ MSE(−), respectively. We also varied γ in E_{SMF}. The values used were {2, 4, 6}, but we found no evident advantage in higher values than $\gamma = 2$.

Table 2 shows the results for selected numbers of hidden units (denoted *hid*) in [2, 20] with the aim of better illustrating the similarities and differences between the several error functions. The results are reported in this way: "mean test error(std. deviation)-epoch". We start by observing that the results of E_{Exp} for $\tau^+ = 100$ and $\tau^- = -100$ are similar or equal to the ones obtained with MSE, which means that E_{Exp} is indeed emulating MSE. Also, by choosing an appropriate value of $\tau \in \mathbb{R}$ and thus, controlling the steepness of the error function, E_{Exp} can perform equally or even better than the CE error function. E_{Exp} also compares favorably to E_{Moller}. We point out the results obtained in New Thyroid: E_{Exp} achieves the best result, improving the solution at least 15% over the other methods (statistically significantly better at 5% probability level). Here, we may also see that the number of epochs used is approximately 3 times smaller than the other error functions. We should also emphasize that the E_{Exp} results exhibit similar or even lower standard deviations than the other methods, which suggests a smaller dependency on the train/test partitions. At the same time, this same aspect is an indication that E_{Exp} is not very sensitive to the choice of optimal τ. As a matter of fact, we noticed in all experiments that a change of τ of roughly 50% had no influence in the results. For instance, for the Ionosphere data set practically the same results were obtained for $\tau \in [1.5, 3]$. However, each data set can have its own best value of τ. The function E_{SMF} also performed well in the two-class problems. It should, however, be extended to the multi-class case to be better evaluated.

6.2. Procedure 2

For data sets with more than 600 cases, a different procedure was applied. Now, the data set is divided in training (50%), validation (25%) and test (25%) sets. For each data set, networks with 2 to 20 hidden units are trained during 1000 epochs. This is repeated 100 times with different initial weights and train/validation/test partitions. At each time, the test set error at the epoch of minimum mean validation error is reported. The same partitions are kept through the different error functions. For this procedure we excluded E_{SMF} as well as E_{Exp} for $\tau^+ = 100$ and $\tau^- = -100$.

Choosing the value of τ and α

The strategy of how to choose the best value for τ and α was the same as above. However, the choice is now ruled by the minimum validation error.

Table 3 shows the results for this experimental procedure as "mean test error(std. deviation)-epoch of minimum validation error". We start by observing that MSE performs badly in some data sets: Pb12, Vowelc and Vehicle. This was found to be caused by several non-convergent runs over the 100 repetitions. On the contrary, the other error functions performed well, achieving the generalization errors reported in the literature for these data sets (see the papers cited in Blake and Merz (1998)). Again, E_{Exp} is capable of achieving the best result among the other error functions and even obtain slight improvements in some cases. For example, in Pb12 for four hidden units or Vehicle with seventeen hidden units, E_{Exp} has the lowest mean error.

Table 2
Results from the application of *Procedure* 1 to six data sets given in the form "%test error(standard deviation)-number of epochs"

hid	Ionosphere $\tau^+ = 2, \tau^- = -10, \alpha = 0.5, \gamma = 4$			Wine $\tau^+ = 5, \tau^- = -10, \alpha = 0.5$		
	2	7	10	2	3	6
MSE	12.60(0.90)-15	12.50(0.89)-11	12.63(0.98)-10	2.37(1.06)-17	2.13(1.00)-16	1.99(0.91)-14
CE	12.26(1.38)-89	12.23(1.28)-81	12.21(1.15)-57	2.64(1.25)-23	2.10(1.03)-20	2.02(0.95)-18
E_{Exp}	12.32(1.38)-63	12.31(1.21)-37	12.30(0.98)-24	2.38(1.11)-15	2.12(0.95)-13	1.96(0.89)-13
E_{Moller}	12.33(1.34)-67	12.29(0.98)-28	12.35(1.04)-28	2.33(1.03)-20	2.12(0.92)-18	1.95(0.88)-18
Z-EDM	12.66(0.89)-17	12.53(0.91)-12	12.66(0.97)-12	2.46(1.01)-22	2.17(1.02)-17	2.01(0.91)-16
E_{SMF}	12.56(1.03)-10	12.54(0.87)-10	12.54(0.89)-9	–	–	–
Exp ↔ MSE(−)	12.59(0.89)-15	12.52(0.90)-11	12.61(0.94)-11	2.35(1.12)-17	2.11(0.98)-16	1.97(0.91)-15
Exp ↔ MSE(+)	12.63(0.95)-15	12.50(0.90)-11	12.61(0.95)-13	2.35(1.01)-17	2.16(0.94)-15	1.97(0.91)-14
hid	New Thyroid $\tau^+ = 0.1, \tau^- = -10, \alpha = 5$			Liver $\tau^+ = 10, \tau^- = -10, \alpha = 0.1, \gamma = 2$		
	2	4	8	17	18	19
MSE	4.69(1.63)-137	4.23(1.40)-126	4.17(1.16)-108	28.89(1.85)-81	28.86(1.89)-83	28.75(1.73)-75
CE	4.26(1.28)-109	4.07(1.06)-44	3.98(1.00)-39	29.37(1.93)-65	29.15(1.91)-66	29.26(1.88)-55
E_{Exp}	3.25(1.04)-48	3.21(0.78)-48	3.22(0.89)-47	28.99(1.84)-84	29.05(1.94)-68	28.76(1.87)-76
E_{Moller}	3.97(1.04)-148	3.80(0.99)-120	3.81(0.98)-129	28.90(1.88)-93	28.94(1.98)-98	28.71(1.84)-90
Z-EDM	5.46(2.64)-200	4.45(2.04)-149	4.39(1.40)-136	29.01(2.09)-97	28.91(1.75)-91	28.87(1.80)-83
E_{SMF}	–	–	–	29.10(1.87)-60	28.87(1.80)-71	28.86(1.78)-76
Exp ↔ MSE(−)	4.73(1.69)-200	4.22(1.38)-135	4.19(1.20)-129	28.81(1.86)-74	28.83(1.93)-89	28.74(1.69)-74
Exp ↔ MSE(+)	4.54(1.55)-186	4.16(1.13)-117	4.13(1.09)-92	28.89(1.83)-77	28.78(1.83)-73	28.94(1.75)-88
hid	Wdbc $\tau^+ = 9, \tau^- = -1.2, \alpha = 0.1, \gamma = 2$			Olive $\tau^+ = 5, \tau^- = -10, \alpha = 0.5$		
	2	3	4	2	3	6
MSE	2.57(0.55)-12	2.56(0.53)-13	2.59(0.53)-13	5.55(0.62)-111	5.41(0.62)-103	5.27(0.57)-108
CE	2.51(0.54)-14	2.58(0.53)-15	2.57(0.50)-14	5.50(0.67)-56	5.37(0.62)-65	5.29(0.65)-56
E_{Exp}	2.57(0.53)-11	2.58(0.51)-12	2.59(0.53)-12	5.39(0.62)-93	5.30(0.59)-104	5.28(0.60)-93
E_{Moller}	2.56(0.53)-27	2.54(0.53)-29	2.56(0.54)-29	5.49(0.67)-113	5.46(0.67)-92	5.26(0.58)-94
Z-EDM	2.49(0.57)-17	2.53(0.58)-19	2.55(0.60)-19	5.61(0.63)-121	5.53(0.72)-96	5.38(0.62)-103
E_{SMF}	2.62(0.56)-10	2.62(0.54)-11	2.64(0.61)-11	–	–	–
Exp ↔ MSE(−)	2.56(0.54)-12	2.56(0.53)-13	2.59(0.55)-14	5.55(0.62)-111	5.41(0.62)-104	5.28(0.60)-113
Exp ↔ MSE(+)	2.58(0.55)-12	2.56(0.53)-13	2.58(0.55)-13	5.55(0.58)-100	5.39(0.64)-90	5.28(0.58)-104

Table 3
Results from the application of *Procedure* 2 to six data sets given in the form "%test error(standard deviation)-number of epochs"

hid	Pima $\tau^+ = 0.5, \tau^- = -10, \alpha = 4$			Pb12 $\tau^+ = 9, \tau^- = -10, \alpha = 0.1$		
	2	3	6	2	4	8
MSE	24.48(3.17)-93	25.09(2.75)-61	25.36(2.83)-46	10.49(8.22)-999	11.66(12.07)-956	13.96(16.62)-990
CE	23.45(2.64)-31	23.61(2.41)-38	23.26(2.67)-34	7.74(1.89)-941	7.54(1.83)-380	7.19(1.70)-486
E_{Exp}	23.36(2.79)-26	23.23(2.79)-22	23.38(2.90)-23	8.02(2.24)-427	7.40(1.80)-642	7.10(1.78)-995
E_{Moller}	23.42(2.78)-67	23.43(3.02)-110	23.30(2.80)-87	7.99(2.54)-937	7.50(1.82)-597	7.17(.189)-743
Z-EDM	23.35(2.74)-67	23.44(2.65)-27	23.22(2.68)-24	8.66(4.50)-974	7.72(1.97)-253	7.56(1.92)-704
hid	Spambase $\tau^+ = 5, \tau^- = -10, \alpha = 0.1$			Vowelc $\tau^+ = 3, \tau^- = -10, \alpha = 2$		
	3	4	6	18	19	20
MSE	7.85(0.77)-942	7.88(0.75)-609	8.07(0.77)-586	35.27(5.41)-1000	35.54(4.95)-991	35.48(6.30)-1000
CE	6.86(0.58)-107	6.77(0.63)-103	6.65(0.64)-141	13.31(2.46)-340	13.00(2.34)-426	12.06(2.50)-372
E_{Exp}	6.86(0.64)-90	6.78(0.61)-129	6.78(0.64)-98	13.10(2.48)-475	12.65(2.61)-514	12.69(2.32)-519
E_{Moller}	6.96(0.73)-306	6.79(0.61)-111	6.72(0.64)-104	13.10(2.48)-475	11.89(2.58)-965	11.53(2.15)-978
Z-EDM	7.04(0.72)-249	6.87(0.64)-105	6.80(0.64)-131	18.16(2.84)-876	17.90(2.81)-989	16.56(3.08)-907
hid	Vehicle $\tau^+ = 4, \tau^- = -10, \alpha = 0.3$			Ctg16 $\tau^+ = 10, \tau^- = -10, \alpha = 0.1$		
	14	17	20	15	18	20
MSE	24.15(2.90)-1000	24.32(3.25)-999	24.65(3.11)-942	21.24(4.03)-990	20.52(2.73)-1000	21.09(2.24)-996
CE	18.57(2.23)-674	18.49(2.60)-743	18.07(2.38)-885	16.51(1.35)-307	16.10(1.47)-400	15.71(1.45)-427
E_{Exp}	18.57(2.44)-954	18.07(2.50)-962	18.29(2.64)-928	15.70(1.38)-982	15.67(1.35)-974	15.59(1.50)-915
E_{Moller}	18.53(2.60)-868	18.53(2.46)-1000	18.22(2.57)-937	16.07(1.48)-998	15.65(1.56)-982	15.50(1.31)-967
Z-EDM	19.06(2.43)-988	18.62(2.71)-988	18.74(2.24)-901	16.33(1.41)-972	16.20(1.65)-988	16.42(1.36)-783

7. Conclusions

We have analyzed the mathematical properties of several error functions with a focus on MLP data classification. This analysis led us to propose two parameterized error functions for MLP training. The first one, E_{SMF}, is a monotonic error function applicable only to two-class problems and was shown to perform equally well as other functions. However, it should be extended to the general multi-class problem for a better evaluation of its capability. The second one, E_{Exp}, an exponential-type error function, constitutes the main contribution of the present paper. This function is able to emulate the behaviors of classic error functions and, as a matter of fact, by single parameter adjustment it is able to implement an infinite family of error functions, with different behavior of the error gradient weighting. This flexible behavior can be valuable in practical applications. The experimental tests described in the paper provide abundant evidence that MLPs trained with E_{Exp} can achieve the best results obtainable with classic error functions and sometimes improve upon them. The E_{Exp} function can be plugged in the usual BP algorithm without increasing the computational complexity, revealing good perspectives for software applications.

Although E_{Exp} was used in data classification with MLPs trained with the BP algorithm, there are no prior reasons precluding its successful use with other training algorithms and/or other types of neural networks in data classification or regression. Several issues concerning theoretical questions (like learning rates, optimality, etc) but also practical ones deserve future attention. A β parameter could eventually be introduced in E_{Exp} as in E_{Moller} and τ could also be made adaptive by analyzing predefined data properties. This would, in principle, not only extend the flexibility of the training process itself but also relieve the user of the choice of τ.

References

Baum, E. B., & Wilczek, F. (1987). Supervised learning of probability distributions by neural networks. In D. Z. Anderson (Ed.), *NIPS* (pp. 52–61). American Institute of Physics.

Bishop, C. (1995). *Neural networks for pattern recognition*. Oxford University Press.

Blake, C., & Merz, C. (1998). UCI repository of machine learning databases. University of California, Irvine, Dept. of Information and Computer Sciences.

Forina, M., & Armanino, C. (1981). Eigenvector projection and simplified non-linear mapping of fatty acid content of italian olive oils. *Annali di Chimica (Rome)*, *50*, 127–155.

Gish, H. (1990). A probabilistic approach to the understanding and training of neural network classifiers. In *Proceedings of the 1990 IEEE international conference on acoustics, speech, and signal processing*.

Hampshire, J., & Waibel, A. (1990). A novel objective function for improved phoneme recognition using time-delay neural networks. *IEEE Transactions on Neural Networks*, *1*(2), 216–228.

Hinton, G. E. (1989). Connectionist learning procedures. *Artificial Intelligence*, *40*(1–3), 185–234.

Jacobs, R., Jordan, M., Nowlan, S., & Hinton, G. (1991). Adaptive mixtures of local experts. *Neural Computation*, *3*, 79–87.

Møller, M. (1993). Efficient training of feed-forward neural networks. *Ph.D. thesis*. Computer Science Department - Aarhus University.

Marques de Sá, J. (2001). *Pattern recognition: Concepts, methods and applications*. Springer Verlag.

Matsuoka, K., & Yi, J. (1991). Backpropagation based on the logarithmic error function and elimination of local minima. In *Proceedings of the 1990 IEEE international joint conference on neural networks*.

Richard, M., & Lippmann, R. (1991). Neural network classifiers estimate bayesian *a posteriori* probabilities. *Neural Computation*, *3*, 461–483.

Santos, J., Alexandre, L., & Marques de Sá, J. (2004). The error entropy minimization algorithm for neural network classification. In *int. conf. on recent advances in soft computing*.

Silva, L., Alexandre, L., & Marques de Sá, J. (2005). Neural network classification: Maximizing zero-error density. In *LNCS: Vol. 3686. ICAPR 2005* (pp. 127–135).

Silva, L., Alexandre, L.A., & Marques de Sá, J. (2006). New developments of the Z-EDM algorithm. In *Proceedings of the sixth international conference intelligent systems design and applications: Vol. 1* (pp. 1067–1072).

Silva, L., Marques de Sá, J., & Alexandre, L. (2005). Neural network classification using Shannon's entropy. In *European symposium on artificial neural networks*.

Silverman, B. (1986). *Density estimation for statistics and data analysis*. Chapman & Hall.

Solla, S. A., Levin, E., & Fleisher, M. (1988). Accelerated learning in layered neural networks. *Complex Systems*, *2*(6), 625–639.