Institute of Phonetic Sciences,
University of Amsterdam,
Proceedings 17 (1993), 1-11.

VOWEL CLASSIFICATION WITH NEURAL NETS: A COMPARISON OF COST FUNCTIONS

David Weenink

Abstract

As part of a larger study on speaker normalization, the merits of some special cost functions for feedforward neural nets are discussed with respect to their classification performance. Both Juang & Katagiri (1992) and Hrycej (1992) claim their cost functions, that are based on minimum classification error (MCE), to be superior to the standard cost function i.e. the one based on a minimum squared error (MSE) criterion. However, no evidence for this claim could be found. On the contrary, for the special condition that a blocked updating scheme for the weights and biases was used in the training procedure, the MCE-based cost functions proved to be inferior to MSE.

1 Introduction

As part of a larger study on speaker normalization, we have been using neural nets to model the process of speaker adaptation (Weenink, 1993). These neural nets were used as special classifiers. The type of cost function that is used during the training phase of a neural net determines to a large extent its classification performance and, therefore, is an essential part of the neural net. Choosing a wrong cost function can have degrading effects. In this paper we want to discuss the merits of cost functions that are based on a criterion called minimum classification error (MCE) in relation to cost functions based on minimum squared error (MSE). In the MSE cost function the classification error is the sum of squares of the differences between the actual outputs and the desired outputs of the neural net. Two recently proposed MCE-based cost functions were developed by Hrycej (1992) and Juang & Katagiri (1992[a]). Especially the latter received considerable attention in the literature (Juang & Katagiri, 1992[b]; Komori & Katagiri, 1992; Kurinami & Sujiyama, 1992). The discussion will be focused on cost functions for supervised feedforward neural nets. Classification capabilities as well as some other aspects of this family of nets have been discussed in Weenink (1991).

For classification, in general, it is not necessary that the output nodes of a neural net contain a nonlinearity. The nonlinear functions that are used for neural nets, for example the sigmoid function, are always monotonic functions. Monotonic functions preserve rank ordering of their inputs. The output node with the largest input produces also the largest output. Consequently, the nonlinearity in the output nodes need not be present. Output nodes that do not contain the nonlinear function are called *linear* since the output of this node is just a linear combination of its inputs.

The search for cost functions is motivated by the following list of shortcomings of the MSE cost function with respect to classification (Hrycej, 1992; Hampshire & Waibel, 1990):

1. The winning class is not identified during learning and is not used in the learning rule either. This means that for classification MSE is not necessarily adequate.
2. The inability to consider an arbitrary cost matrix. In MSE classification it is not possible to consider an individually specified cost for each misclassification type, i.e. the cost for classifying a member of the i-th class as a member of the j-th class.
3. MSE, in combination with linear outputs, slows down the convergence of learning by overconstraining the problem. Generally the desired output of the correct class is given the value 1 and the other desired outputs the value 0. It may be clear that a value greater than 1 for the correct output class and a value smaller than 0 for the incorrect classes would be harmless. However, MSE penalizes such cases and it can be expected that these unnecessary constraints slow down the learning process.
4. The MSE cost function is not monotonic with respect to classification when the number of classes (M) exceeds one. In other words, patterns with a 'low' MSE may be classified wrongly while patterns with a 'high' MSE may be classified correctly. E.g. the maximum MSE for a correctly classified pattern could have, besides the correct unit being 1, all other (wrong) outputs near 1. In this case the MSE is approximately equal to M-1. The minimum MSE for a wrongly classified pattern could have, besides the correct unit being 0, all other (wrong) outputs near 0, giving rise to MSE=1.

Before we discuss the merits of the alternatives we will have a look at how the cost function influences network parameters.

2 The relation between cost function and weights

Before a neural net can be used as a classifier it has to be trained. The purpose of training is to obtain a set of weights and biases that minimizes a certain cost function E over the training set. Training a one-layer net is very simple since the desired output is known and the only weights are those between the outputs and the inputs. This means that weights can be gradually updated until the error is sufficiently small. However, this procedure is not directly applicable to a net with hidden nodes because most of the time one does not know what the hidden nodes should represent: there is no desired output for the hidden nodes. Despite this problem, successful training procedures for nets with hidden layers have been developed. The most common learning algorithm uses a gradient search technique to find the network weights and biases $\mathbf{w}$ that minimize the cost function $E(\mathbf{w})$. It is called the back propagation algorithm (Rumelhart et al., 1986). The weights and biases of the network are determined iteratively until a minimum of $E(\mathbf{w})$ is found according to:

$$w_{l,ij}(k+1) = w_{l,ij}(k) - \mu \frac{\partial E(\mathbf{w})}{\partial w_{l,ij}} \tag{1}$$

where μ is a positive constant, called the learning rate. To minimize the cost function $E(\mathbf{w})$, we have to derive an expression for the partial derivative of the error function

with respect to each individual weight in the network. Before we can do so we have to define some terms that will be used in the derivation. For a node j in layer l, the output in relation to its inputs is:

$$O_{l,j} = f_l\left(I_{l,j}\right) \tag{2}$$

$$I_{l,j} = \sum_{k=1}^{N_{l-1}} w_{l,kj} O_{l-1,k} \tag{3}$$

Here $O_{l,j}$ is the output of the node j in the layer l, $w_{l,kj}$ is the weight that connects node k in layer l-1 with the j-th node in layer l, $I_{l,j}$ is the input of the nonlinearity f_l of the j-th node in layer l , and N_{l-1} the number of node at layer l–1. We define the error at node j of layer l as follows:

$$\delta_{l,j} = -\left(\frac{\partial E}{\partial I_{l,j}}\right) \tag{4}$$

When we use the chain rule on (4) we get:

$$\delta_{l,j} = -\left(\frac{\partial E}{\partial I_{l,j}}\right) = -\sum_{k=1}^{N_{l+1}} \frac{\partial E}{\partial I_{l+1,k}} \cdot \frac{\partial I_{l+1,k}}{\partial I_{l,j}} = \sum_{k=1}^{N_{l+1}} \delta_{l+1,k} \cdot \frac{\partial I_{l+1,k}}{\partial I_{l,j}}$$

The last term in the summation can be simplified as:

$$\frac{\partial I_{l+1,k}}{\partial I_{l,j}} = \frac{\partial}{\partial I_{l,j}} \sum_{p=1}^{N_{l+1}} w_{l+1,pk} O_{l,k} = \frac{\partial}{\partial I_{l,j}} \sum_{p=1}^{N_{l+1}} w_{l+1,pk} f_l\left(I_{l,p}\right) = w_{l+1,jk} f_l'\left(I_{l,j}\right)$$

The two equations above then combine to:

$$\delta_{l,j} = f_l'\left(I_{l,j}\right) \sum_{k=1}^{N_{l+1}} w_{l+1,kj} \delta_{l+1,k} \tag{5}$$

Relation (5) expresses the *back propagation* of errors. The errors δ_l at the lower layer l can be calculated from the errors at the next higher layer l+1. The derivative of the cost function with respect to the weights can now simply be written down as:

$$\frac{\partial E}{\partial w_{l,ij}} = \frac{\partial E}{\partial I_{l,j}} \cdot \frac{\partial I_{l,j}}{\partial w_{l,ij}} = -\delta_{l,j} O_{l-1,i} \tag{6}$$

The attractiveness of this formulation of the derivative lies in the fact that in (6) no explicit notion of the cost function figures any more. Derivative information at a layer l is expressed in terms of δ_l and O_l. The specifics of the cost function only enter at the top layer.

When we minimize the errors between the desired outputs and the actual outputs of the net in a quadratic sense, it is called the Minimum Squared Error (MSE) criterion function.

$$E(\mathbf{w}) = \frac{1}{2}\sum_{j=1}^{M}\left(O_j(\mathbf{p}) - d_j(\mathbf{p})\right)^2 \tag{7}$$

Here $O_j(\mathbf{p})$ denotes the output of the j-th output node of the neural net for pattern $\mathbf{p}$ and $d_j(\mathbf{p})$ the desired output for this pattern on this node. M denotes the number of outputs. This cost function is at a minimum when for all patterns for all output nodes the output of the net is equal to the desired output. For this cost function, the errors at the top level, that propagate back, can simply be calculated according to formula (4) as:

$$\delta_{L,j} = f'_L\left(I_{L,j}(\mathbf{p})\right)\cdot\left(O_j(\mathbf{p}) - d_j(\mathbf{p})\right) \tag{8}$$

Two schemes for updating the weights and biases exist, *incremental* and *blocked* updating. In the incremental updating scheme weights and biases are changed after each training pattern. One usually uses this scheme in adaptive sessions when the total training set is not available or continuously changing. When the total training set is fixed a *blocked* updating of the weights is more favourable. In this case the weights and biases are updated only after each complete iteration of all training patterns. Usually this is faster than updating after each training sample. Furthermore, blocked updating has better convergence properties because the cumulative gradient, which is a mean gradient over the training set, converges to zero for the optimal parameter values.

In the minimization of the cost function we can use gradient information in a special way. The standard minimization method, as described by equation (1), is the steepest descent method. In this method weight changes are always in the direction of the gradient. This method leads to a not very good algorithm of minimization. The problem with the steepest descent method is that it will perform many small steps in going down a long, narrow valley, even if the valley has a perfect quadratic form. Because the new gradient at the minimum point of any line minimization is perpendicular to the direction just traversed, you always must make a right angle turn, which does not, in general, take you to the minimum. Instead we want a way of proceeding not down the new gradient, but rather in a direction that is somehow constructed to be conjugate to the old gradient and previous directions. Conjugate gradient methods accomplish this and therefore are, under many circumstances, superior to steepest descent methods. In our neural net simulation program we have implemented three different minimization algorithms, two of which are based on conjugate gradients. The simplest minimization method implemented is steepest descent with an (optional) momentum term. The second, most powerful, method is Powell's conjugate gradient method (Van der Smagt & Kröse, 1991). The third method is the Fletcher-Reeves-Polak-Ribiere conjugate gradient method. An introduction to conjugate gradient methods of minimization can be found in Press et al. (1992).

In the initial phase of the training the sizes of the random weights and the inputs are essential. When they are too large, the weighted sums of the inputs, which form the inputs to the sigmoid nonlinearities, can be relatively big numbers. As a consequence, the training algorithm starts where the sigmoid functions are at a position where the derivatives are extremely small, and, since the speed of updating weights is a function of this derivative, hardly any training of the net is the effect. A sensible strategy to avoid this is to choose the initial random weights so that the magnitude of the typical input to a unit is somewhat less than unity. This can be achieved by initializing the weights in a layer to a random number in the interval

$(-n^{-\frac{1}{2}}, n^{-\frac{1}{2}})$, where n is the number of units which feedforward to this layer. In an analogous way the inputs, when they are too large, can block the training process. The remedy is scaling the inputs to values located around zero, say in the interval (-1, +1) or (0, 1).

3 The cost function of Juang & Katagiri

The first alternative to MSE that we consider was formulated by Juang & Katagiri (1992[a]). It contains explicit notion of the winning class and therefore addresses the first point of the list 'shortcomings of MSE' in section 1. They define the cost function for pattern $\mathbf{p}$ to be a sigmoid function of a continuous misclassification measure d_k:

$$E(\mathbf{p}) = \sigma(d_k(\mathbf{p})) = \frac{1}{1+e^{-d_k(\mathbf{p})}} \tag{9}$$

where d_k is defined as:

$$d_k(\mathbf{p}) = -O_k(\mathbf{p}) + \left[\frac{1}{M-1}\sum_{j,j\neq k} O_j^{\eta}(\mathbf{p})\right]^{\frac{1}{\eta}} \tag{10}$$

Here pattern $\mathbf{p}$ is supposed to belong to class k (the correct output class), $O_k(\mathbf{p})$ is the output of the correct node of class k resulting from input pattern $\mathbf{p}$, η is a positive number and there are M classes. In this formula (10), the correct class appears explicitly (via $O_k(\mathbf{p})$) and the incorrect classes enter in a weighted sum (the term enclosed by the square brackets). When η approaches ∞, the misclassification measure becomes:

$$d_k(\mathbf{p}) = -O_k(\mathbf{p}) + \max_{i\neq k} O_i(\mathbf{p}) \tag{11}$$

where i is the index of the class with the largest output value other than the correct class k. It is clear that in this case $d_k > 0$ implies misclassification and $d_k \leq 0$ means correct decision. The errors at the *linear* output nodes, according to equation (4), then become:

$$\begin{aligned}\delta_{L,j} &= -\left(\frac{\partial E}{\partial I_{L,j}}\right) = -\left(\frac{\partial E}{\partial O_{L,j}}\right) = -\frac{\partial \sigma(d_k)}{\partial d_k}\sigma(d_k)\frac{\partial d_k}{\partial O_{L,j}} \\ &= -\sigma(d_k)(1-\sigma(d_k))\frac{\partial d_k}{\partial O_{L,j}}\end{aligned} \tag{12}$$

where the argument pattern vector $\mathbf{p}$ is implied, L is the index of the output layer, and,

$$\frac{\partial d_k}{\partial O_{L,j}} = \begin{cases} -1, & j = k \\ \dfrac{O_{L,j}^{\eta-1}}{M-1}\left\{\dfrac{1}{M-1}\displaystyle\sum_{i,i\neq k} O_{L,i}^{\eta}\right\}^{-1+1/\eta}, & j \neq k \end{cases} \tag{13}$$

which, for $\eta \to \infty$, becomes:

$$\frac{\partial d_k}{\partial O_{L,j}} = \begin{cases} -1, & j = k \\ 1, & j = \text{argmax}_i(O_i) \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

We emphasise again that in equations (12-14) it is understood that the output units are linear, i.e. no sigmoid function is active in these units.

In all subsequent comparisons of classification performances between the two cost functions the training data set and the test data set were the same. There were two data sets for which we compared classification performances: Fisher's iris data set and van Nierop et al.'s female vowel formant data. The iris data consist of four measurements made by E. Anderson (1935) on 150 samples of three species of iris. The four measurements are sepal length, sepal width, petal length and petal width. Fifty tokens are available for each of the three classes. The iris data set was one of the test sets used by the authors of this MCE-based cost function and is used extensively in the literature as a reference set (Fisher, 1936). However, we must note that it is not a very interesting data set for classification since standard linear discriminant analysis with the program SYSTAT (Wilkinson, 1989) already gives a classification rate of 98.0%, only 3 out of 150 are being misclassified. This means that the misclassification of 2.2% that Juang and Katagiri obtain by their formula (9) is not impressive (their table III). In fact with our neural net simulation program we easily reach 0.7% misclassification with a one-layer neural net of topology (4, 3), i.e. 4 inputs, 3 outputs, no hidden units, a sigmoid nonlinearity, and the MSE cost function. This contrasts heavily to their 12.3% misclassification, which was simulated with a net of topology (4, 15, 3). With MSE, 15 hidden units, and a *nonlinear* output layer we obtained 100% correct classification. The iris data set does not show the superiority of the MCE cost function as Juang & Katagiri argue. On the contrary, there is slight superiority for the MSE based cost function.

The principal weakness of this MCE-based cost function will reveal itself with a data set that needs considerable more output classes than the iris data set such as the formant frequency measurements of 25 female speakers of van Nierop et al. (1973). The van Nierop et al. set consists of the first three formant frequencies in Hertz of the 12 Dutch vowels (/u/, /ɔ/, /o/, /ɑ/, /a/, /ʏ/, /ø/, /y/, /i/, /ɪ/, /e/ and /ɛ/). It consequently needs 12 output classes. These formant frequency values were scaled to values in the interval (0, 1) according to the following formula:

$$F_i' = \frac{f(F_i) - f(F_{i,\min})}{f(F_{i,\max}) - f(F_{i,\min})} \tag{15}$$

Since all vowel classifications in the Van Nierop et al. paper were performed on log$F$ values, the function f was chosen to perform a logarithmic formant frequency transformation by taking $f(x) = \ln(x)$. This transformation is then followed by a linear scaling. The following values were used for the parameters of this linear scaling: the minimum formant frequency values, $F_{i,\min}$ for the first three formants were chosen to be 200, 500, and 1500 Hz, respectively. The maximum formant frequency values, $F_{i,\max}$, were 1500, 3500, and 4500 Hz, respectively. This linear scaling makes logarithmic scaling independent of the base of the logarithm and has the additional

advantage that all transformed frequencies are in the range (0, 1) which guarantees better training.

With MSE we get excellent classification on the van Nierop et al. data set as is shown in the last column of table 1. When the MCE cost function (9) in combination with (10) was used for *finite* η, classification results were generally worse. We only present results here, and not in table 1, for a net of topology (3, 10, 12). The percentage correct on the average was less than 70% correct with the following settings of the simulation program: weights were initialized at a random value between 0.1 and -0.1, a blocked update scheme was used with conjugate gradient minimization, η=4 was chosen in formula (10), linear output nodes were used, the number of iterations was chosen sufficiently large (>10,000) to guarantee good minimization. The performance of less than 70% correct classification is substantially below the 88.3% correct classification obtained with the combination of MSE and nonlinear output nodes. Apart from the worse classification performance, we notice that the blocked minimization with MCE more often got stuck in a local minimum than MSE minimization. Using a pattern by pattern update and choosing appropriate values for μ and α did not help. A careful look at the patterns that were not correctly classified revealed a flaw in this MCE-based cost function. For finite values of η, the value of d_k can become very negative, meaning very low cost, even when the output value of the correct class is very much smaller than that of one (or more) incorrect class(es). The derivative of the cost function, equation (13), is very small too in this situation, meaning that virtually no correction on this unfavourable situation is taking place. As long as the average of the M-1 values of O_j^η stays much below the value O_k^η, the misclassification does not add much to the cost function. Indeed, the total cost function can reach any small value ε ($\varepsilon > 0$) without 100% classification, a very undesirable property. For example, in one session with MCE simulation as above, the total cost was minimized from an initial value of 250.0 to a value of 0.0013 with only 67.3% correct identification. In another session the total cost was reduced from an initial 150.0 to a final 0.00038 and, despite a reduction of the cost with a factor of 10^5, only 58.3% correct classification resulted. When the number of classes (M) increases, the probability that this phenomenon occurs is likely to increase.

Table 1. Comparison of classification performance between MSE cost function and MCE cost function of *Juang & Katagiri* with $\eta \to \infty$. The topology of the neural net was (3, N, 12). The training data set of van Nierop et al. was used (see text). MCE was tested with pattern-by-pattern update with randomization (μ=0.003 and α=0.9). The columns, from left to right, denote the number of hidden units, the MCE-cost after training, and the percentages correct for MCE- and MSE-based training, respectively. The percentage correct derived in the van Nierop et al. study via maximum likelihood classification was 79.0%.

#Hidden	Cost	MCE (%)	MSE (%)
2	72.0	77.0	75.6
3	65.0	78.0	79.3
4	58.2	81.3	80.3
5	55.9	82.0	83.0
6	53.7	83.0	82.6
7	52.5	83.0	86.6

Since finite η does not do the job, the only formulation of this MCE cost function that needs checking is the limiting case $\eta \to \infty$: only the difference of the output for

the correct class and the highest output of the resulting output units appears in the cost function. The measure in equation (11) for d_k now clearly is better coupled to classification performance than before: a negative value meaning correct classification. In table 1 we have accumulated some results of testing this MCE-based cost function. We have tested MCE by updating the weights after each pattern was clamped, for each iteration the patterns were randomized. The values chosen for the gradient descent were a pattern-by-pattern update with μ=0.003 and α=0.9. The number of iterations was chosen to be sufficiently high (50,000). We had to use this pattern-by-pattern updating scheme because the blocked updating scheme did not perform reliably with this cost function, it got stuck many times in a local minimum. It seems that with MCE in the blocked update case, in one way or another, many times a heavy cancellation of weight changes is taking place, in such a way that no effective updating is possible any more. The results in table 1 show that with the van Nierop et al. data set the results, especially for topologies with a small number of hidden units, are satisfying. But these results come at great cost: many times the blocked updating of weights with fast conjugate gradient minimization cannot be used, pattern by pattern updating with steepest descent has to be used instead. Furthermore, the learning parameters (μ, α) have to be optimally adjusted to guarantee proper minimization and patterns have to be randomized in each iteration. Pattern-by-pattern updating takes considerable more computer time than a blocked update. Moreover, our powerful minimization algorithms cannot be used. Careful testing with more difficult artificially generated data sets with strongly overlapping classes showed clear superiority of the MSE cost function in combination with nonlinear output nodes over MCE with linear output nodes. In summary, we could not find convincing evidence for the superiority of this MCE-based cost function over the standard MSE-based cost function.

4 The cost function of Hrycej

The cost function of Hrycej (1992) is the simplest form of a function that imposes no weight changes if classification is correct. It has a non-zero gradient only in the region were the cost is positive. The cost function is:

$$E(\mathbf{p}) = C_{ki}\,\mathrm{pos}\big(O_i(\mathbf{p}) - O_k(\mathbf{p})\big) \tag{16}$$

in which k is the index of the correct class of pattern $\mathbf{p}$, i the index of the largest output, C_{ki} is the element of the cost matrix that denotes the cost of misclassifying a pattern belonging to the correct class k as belonging to the incorrect class i, and the function pos(u) is defined as pos(u)=u for $u > 0$ and pos(u)=0 otherwise. The cost matrix C_{ki} need not be symmetric and can be any general matrix. The errors at the output level can be expressed as

$$\delta_{L,j} = \begin{cases} -C_{ki}\theta(O_i - O_k), & j = k \\ C_{ki}\theta(O_i - O_k), & j = i \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

where $\theta(u)$ denotes the step function, defined by $\theta(u)$=1 if $u > 0$ and $\theta(u)$=0 otherwise. This cost function, in combination with blocked updating, has the admirable property that it is a convex function with regard to the classifier

parameters, i.e. the weights (Hrycej, 1992). This means that the global minimum of the cumulative cost function can be found by gradient descent.

In table 2 we have accumulated some results of the comparison of MCE versus MSE. The data set used for the comparison was van Nierop et al.'s vowel formant frequency data set of 25 Dutch female speakers. We mention that, as was the case with the MCE-based cost function of the previous section, a blocked updating scheme of the weights was not very successful. Again, we had to use incremental updating. We chose μ=0.003 and α=0.9. The classification results for this MCE-based cost function were not very impressive. We did several other tests with data sets with strongly overlapping classes and this MCE cost function did not perform well. Many times it got stuck in a local minimum without any substantial classification performance. Careful analysis of the resulting states lead us to detect a defect in this cost function: The main weakness of the Hrycej cost function is that it is too sensitive to scale: a trivial reduction of all output weights and biases with a factor α ($0 < \alpha < 1$) reduces the outputs with the same factor because the output nodes are linear. The net effect of this reduction is that the total cost is reduced with the same factor, however, without any implication on the classification performance whatsoever. The global cost can be reduced to any number ε ($\varepsilon > 0$), without affecting the classification at all. A very undesirable property for a cost function.

Table 2. Comparison of classification performance between MSE cost function and MCE cost function of *Hrycej*. Incremental updating scheme with μ=0.003 and α=0.9. For further details see table 1.

# Hidden	Cost	MCE (%)	MSE (%)
2	0.43	72.0	75.6
3	0.42	74.3	79.3
4	0.34	73.0	80.3
5	0.24	73.3	83.0
6	0.44	71.3	82.6
7	0.86	72.0	86.6

5 Discussion on cost functions

Most of the criticism formulated in section 1 on MSE-based learning is only appropriate for the combination of MSE and linear output units. The only serious objection that MSE has no remedy for, is the first one of the list: the MSE cost function is not necessary optimal for classification. The rest of the objections can be dealt with easily as we will demonstrate.

A class specific cost can be incorporated in the MSE cost function of (7) in an analogous way as was done in the previous section with the Hrycej cost function:

$$E(\mathbf{p}) = \frac{1}{2}\sum_{i=1}^{M} C_{ki} \cdot (O_i - d_i)^2 \tag{18}$$

In this cost function, pattern $\mathbf{p}$ belongs to class k and C_{ki} is the cost when class k is misclassified as class i. This reduces to the standard MSE formulation when all C_{ki} are equal to 1. The errors at the output level can simply be calculated:

$$\delta_{L,j} = f'_L\left(I_{L,j}\right) \cdot C_{kj} \cdot \left(O_j - d_j\right) \qquad (19)$$

where f is the function present at the output nodes. As was explained in section 1, when used in combination with linear units the MSE cost function slows down learning. However, this need not worry us since MSE and linear output nodes were not meant for each other. The solution is to change the output function to a sigmoid function. This immediately creates the necessary freedom and removes the unnecessary constraints because the domain of the sigmoid is $(-\infty, +\infty)$. The price we have to pay for changing the output function to a sigmoid is an increase in the learning time. All MSE classification tests in this paper were performed with sigmoid nonlinearities in *all* nodes.

A cost function that is monotonic with respect to classification cannot be a function of all outputs at the same time. When the number of classes is substantial and the cost function is non-monotonic it is always possible that misclassified patterns with low cost exist. It may be that for these misclassified patterns the way to go in weight space in order to reach perfect classification is either partly uphill or very slowly downhill. In the averaging performed by a cumulative update this can normally be remedied.

The only objection against MSE that remains valid is the first argument in the list of section 1, MSE is not necessarily optimal for classification.

From the discussion above we must, however, conclude that, although MSE as a cost function is probably suboptimal, it is certainly hard to beat.

Acknowledgement

Part of this work was carried out on an RS/6000 computer which was kindly provided by IBM under the terms of the ACIS programme.

References

Anderson, E. (1935), "The irises of Gaspé Peninsula", *Bulletin of the American Iris Society* **59**, 2-5.

Fisher, R.A. (1936), "The use of multiple measurements in taxonomic problems", *Ann. Eugenics*, part II, vol. 7, 179-188.

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

Hrycej, T. (1992), *Modular learning in neural networks: a modularized approach to neural network classification*, John Wiley & Sons, Inc.

Juang, B. H. & Katagiri, S. (1992[a]), "Discriminative learning for minimum error classification", *IEEE Trans. on Speech Proc.* **40**, 3043-3054.

Juang, B. H. & Katagiri, S. (1992[b]), "Discriminative training", *J. Acoust. Soc. Jpn (E)* **13**, 333-339.

Komori, T. & Katagiri, S. (1992), "GPD training of dynamic programming-based speech recognizers", *J. Acoust. Soc. Jpn (E)* **13**, 341-349.

Kurinami, K. & Sujiyama, M. (1992), "An optimization technique for speaker mapping neural networks using minimal classification error criterion", *J. Acoust. Soc. Jpn (E)* **13**, 419-427.

Press, W.H., Flannery, B.P., Teukolsky, S.A. & Vetterling, W.T. (1992), *Numerical recipes in C: The art of scientific computing*, Cambridge University Press.

Rumelhart, D.E., Hinton, G.E. & Williams, R.J. (1986), "Learning internal representations by error propagation", reprint in: *Neurocomputing: Foundations of research*, Anderson, J.A. and Rosenfeld, E. (eds.), MIT Press, Cambridge: 675-695.

Van der Smagt, P.P. & Kröse, B.J.A. (1991), "A real-time learning neural robot controller", *Proceedings of the International conference on Artificial Neural Networks*, Espoo, Finland.

Van Nierop, D.J.P.J, Pols, L.C.W. & Plomp, R. (1973), "Frequency analysis of Dutch vowels from 25 female speakers", *Acustica* **29**, 110-118.
Weenink, D.J.M. (1991), "Aspects of neural nets", *Proceedings of the Institute of Phonetic Sciences Amsterdam* **15**, 1-25.
Weenink, D.J.M. (1993), "Modelling speaker normalization by adapting the bias in a neural net", *Proceedings Eurospeech '93*, 2259-2262.
Wilkinson, L. (1989), *SYSTAT: The system for statistics*, Evanston, IL, Systat, Inc.