Pattern Anal Applic (2003) 6: 201–211
DOI 10.1007/s10044-002-0188-0

ORIGINAL ARTICLE

J. A. Bilmes and K. Kirchhoff

Generalized rules for combination and joint training of classifiers

Received: 3 November 2001 / Accepted: 17 May 2002

Abstract Classifier combination has repeatedly been shown to provide significant improvements in performance for a wide range of classification tasks. In this paper, we focus on the problem of combining probability distributions generated by different classifiers. Specifically, we present a set of new combination rules that generalize the most commonly used combination functions, such as the mean, product, *min*, and *max* operations. These new rules have continuous and differentiable forms, and can thus not only be used for combination of independently trained classifiers, but also as objective functions in a joint classifier training scheme. We evaluate both of these schemes by applying them to the combination of phone classifiers in a speech recognition system. We find a significant performance improvement over previously used combination schemes when jointly training and combining multiple systems using a generalization of the product rule.

Keywords Classifier combination · Combination rules · Joint classifier training · Mixtures of experts · Neural network ensembles · Neural networks · Products of experts · Speech recognition

Introduction

An attractive approach to pattern recognition which has received much attention is classifier combination [1]. The underlying goal of classifier combination research is to identify the conditions under which the combination of an ensemble of classifiers yields improved performance compared to the individual classifiers. Several empirical and theoretical studies [2–5] have found that combination is most successful when the errors of the ensemble members are as uncorrelated as possible. Producing ensemble members with decorrelated errors can be achieved by a variety of methods, e.g. by training classifiers with different structures or varying the initial conditions from which classifiers are trained.

Many studies on classifier combination have focused on what we call single-level classification. A typical single-level classification system consists of two components: a feature-extraction component which maps the input signal to feature vectors, and a classification component which assigns a class label to each feature vector. The desired classes are thus directly recognized from the feature space without any intermediate representation. The classification component may involve a combination of several different classifiers, in which case the optimal combination method is the one which yields the highest classification accuracy on a test set, or in other words, which produces the classifier which generalizes best to unseen data.

Many real-world pattern recognition applications, however, require a multi-level classification procedure. In such a case, the class label is related to the features indirectly via some intermediate representation. For example, a second level classification scheme might base its decisions on the result of the first level classification scheme. Classifier combination in a multi-level classification system must take into account the properties of this intermediate representation and might therefore differ greatly from those combination methods which have proved successful for single-level classification.

One example of a multi-level classification system is Automatic Speech Recognition (ASR). Except for systems dealing with an extremely limited recognition vocabulary (e.g. digits), ASR systems typically do *not* recognise words directly from the preprocessed speech signal – words are represented in terms of smaller, intermediate units, such as phones.[1] Phones are classified directly from the speech signal; their scores are subsequently passed on to a higher-level *search* component which selects the best

J. A. Bilmes ✉
Department of Electrical Engineering, University of Washington, EE/CS, Box 352500, Seattle, WA, USA. Email: bilmes@ssil.ee.washington.edu

K. Kirchhoff
SSLI Laboratory, Department of Electrical Engineering, University of Washington, Seattle, WA, USA

[1] Other units can also be used such as syllables, demi-syllables, etc.

path from the space of possible phone sequences. The search component is often called Viterbi search, where only the best path over time through the sequence of phonemes is obtained by a dynamic programming recursion.

In recent years, classifier combination has been attempted on the speech recognition task because of the potentially large benefits to be gained. Robust speech recognition is a difficult task due to the variable nature of the speech signal. Signal bandwidth, background noise and speaker variability (such as different speaking rates and foreign accents) are all examples of conditions which severely affect recognition accuracy. It is highly unlikely that a monolithic ASR system using a single acoustic classifier will exhibit consistently high performance across a variety of recognition conditions.

In this paper, we address the problem of combining classifiers in multi-level classification systems with application to phone classifier combination in a speech recognition system. Specifically, we present a set of new combination rules which are shown to be generalizations of the well-known average, product, *min* and *max* rules. These new combination rules have continuous and differentiable forms. Therefore, they can not only be used for the combination of separately trained classifiers, but they can also be used as objective functions in a joint classifier training scheme. We discuss the properties of joint training and present preliminary results with respect to word recognition. While the techniques in this paper are applied to multi-level classification and to speech recognition, our methods are quite general and could be applied to any form of classifier combination methodology for which combination rules (such as *min*, sum, and so on) are appropriate. We explore this case later in the paper when we empirically analyse 4-class two-dimensional Gaussian data.

The rest of this paper is organised as follows: in Sect. 2 we briefly review previous work on classifier combination in both the general machine learning field and in automatic speech recognition. In Sect. 3 and 4 we present new combination rules that generalize almost all of the existing linear combination rules proposed in the classifier combination literature. In Sect. 5 we provide the derivation of a joint classifier training scheme using these new rules. Experimental results are presented and discussed in Sect. 6, relationships to other work are provided in Sect. 7, and a summary is given in Sect. 8.

Classifer combination in machine learning

The goal of classifier combination research is to identify the conditions under which an ensemble of classifiers yields the largest gain in performance compared to the individual classifiers. One widely investigated combination method is the mean rule [1], i.e. a weighted average of the outputs of the individual classifiers:

$$P(c|x_1, \ldots, x_N) = \sum_{n=1}^{N} \alpha_n P(c|x_n) \tag{1}$$

where x_n is a feature vector, $P(c|x_n)$ is the probability for class c given by the nth classifier, and α_n is the weight for the nth classifier.

Classification is closely related to regression, and several theoretical studies [2,7] have shown that mean-rule combination is successful (i.e. has a lower mean-squared error) when the errors of each system are independent. Error reduction is related to ensemble bias (the degree to which the averaged output of the ensemble of classifiers diverges from the true target function) and variance (the degree to which the ensemble members disagree) [3,6,7]. Generally, a low error requires both a low bias and variance, but since variance is reduced by averaging over a greater number of classifiers, it is sufficient to combine classifiers with a low bias. Tumer and Ghosh [8] have related the degree of correlation of classifier outputs to the ensemble error, and have quantified it in terms of Bayes error. The total ensemble classification error E_t can be represented as the Bayes error E_b and the added ensemble-incurred error $\bar{E}_a$, having the relationship $E_t = E_b + \bar{E}_a$. When combining unbiased correlated classifiers, the added error can be shown to be

$$\bar{E}_a = E_a \frac{1 + \rho(N - 1)}{N} \tag{2}$$

where N is the number of classifiers, ρ a measure of error correlation, and E_a is the (common) added error of each individual classifier. As can be seen, the added error grows with the degree of classifier error correlation.

Producing ensemble members with decorrelated errors can be achieved by a variety of methods, e.g. by training classifiers with different structures [8], varying the initial conditions from which classifiers are trained, training on disjoint [9] or partially overlapping data sets, using different input signals [9], or different feature representations of the input [10]. Popular combination methods include linear combinations of the output distributions (e.g. by averaging over, or multiplying, the outputs) [1,11], combining the outputs by a higher-level classifier, Dempster-Shafer theory [12], and majority voting [13]. An alternative approach is to explicitly model the dependence between classification errors [7,8,14].

Many studies on classifier combination focus on what we call single-level classification. A typical single-level classification system consists of two components: a feature-extraction component, which maps the input signal to feature vectors, and a classification component, which assigns a class label to each feature vector. The desired classes are thus directly recognised from the feature space without using any intermediate representation.

There are many pattern recognition tasks, however, that require a multi-level classification scheme because, for practical reasons, it is necessary to reuse classifier results at a middle level. Essentially, the final resulting classes are encoded using an intermediary class representation. A class random variable C is related to the features $x_1, \ldots, x_N$ indirectly via the intermediate representation Q, and the system often makes simplifying conditional independence assumptions as in:

$$p(c|x_1, \ldots, x_N) = \sum_q p(c, q|x_1, \ldots, x_N)$$
$$= \sum_q p(c|q)p(q|x_1, \ldots, x_N)$$

where Q itself can be a multi-dimensional vector, and which itself might use a multi-level classification scheme. Classifier combination can be applied at any of the intermediate stages in a multi-level classification system. The combination methods, however, should ideally take into account the requirements of the last stage (i.e. the Bayes error for the final desired class variable C). Successful combination methods in a multi-level classification system therefore might differ greatly from those combination methods that have proved beneficial for single-level classification.

Classifier combination in speech recognition

In speech recognition, it is more efficient for the class C (which is often a sentence) to make use of a relatively compact set of classifiers Q representing sub-phones, phones, or even words, rather than attempting to discriminate directly between what would be an inordinate number of classes (i.e. the number of possible sentences). The interaction between the different levels crucially determines the overall performance and creates unique conditions for classifier combination which differ from single-level classification.

The goal of ASR is to extract the most likely word sequence $\hat{W} = \{w_1, w_2, \ldots, w_N\}$ from a preprocessed speech signal, i.e. sequence of observation vectors $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_T$. This is done according to Bayes Rule:

$$\hat{W} = argmax_W \frac{P(\mathbf{X}|W)P(W)}{P(\mathbf{X})} \tag{3}$$

The prior probability of the word sequence, $P(W)$, is approximated by a product of the conditional probabilities of the individual words in the sequence given their preceding words (the *language model*). These quantities are computed in advance on a training set and shall not concern us any further here. The likelihood $P(\mathbf{X}|W)$ is accumulated during recognition by the acoustic classification component, which we focus on in this study. As mentioned before, $P(\mathbf{X}|W)$ is not computed directly on the word sequence. Rather, words are represented as sequences of smaller units, so-called *phones*, which represent individual speech sounds. The word sequence W can thus be expressed as a phone sequence $Q = q_1, q_2, \ldots, q_M$. The acoustic classification component then computes, for each speech frame, the likelihood of a given feature vector x given a statistical model for any given phone q, $P(\mathbf{x}|q)$. In most ASR systems $P(\mathbf{x}|q)$ is modelled by a mixture of Gaussian densities; however, Artificial Neural Networks (ANNs) can also be used [15]. In this study we employ Multi-Layer Perceptrons (MLPs) to estimate $P(q|\mathbf{x})$. For the purpose of decoding, $P(q|\mathbf{x})$ is converted into the required (scaled) likelihood $P(\mathbf{x}|q)$ by dividing by the prior probability $P(q)$.

Different partial recognition hypotheses can in principle be combined at either the phone, word, or sentence level. In this study, we concentrate on combination at the phone level. Most approaches to phone-level combination have used different acoustic preprocessing techniques to generate an ensemble of classifiers trained on different feature spaces [16–18]; in some cases, the speech signal is split into a number of narrower frequency bands and the ensemble classifiers operate on individual sub-bands [19,20]. Combination methods have in general used either the mean rule (Eq. (1)) or the product rule.

$$P(c|x_1, \ldots, x_N) = \frac{\prod_{n=1}^{N} P(c|x_n)}{Z} \tag{4}$$

where Z is a normalising constant. The mean rule is useful for combining unimodal distributions into a single multimodal distribution. Since mixing can only increase entropy of a distribution [22] relative to the mixture of the individual distribution entropies, such a procedure is poor for representing low-entropy distributions where probability is concentrated in narrow input-space regions. In such cases, the product rule is useful, where each classifier must supply probability to the correct class, but may also supply probability to incorrect classes as long as one or more of the other classifiers do not supply probability to those incorrect classes. These are therefore called ‘AND’ style combination schemes [23] since only the logical AND of each classifier’s probabilistic decision will survive combination. It is also the case that such a combination scheme is useful when the underlying distributions factorise over the probabilistic space of C [24].

Virtually all the aforementioned studies reported the largest performance increases for the product rule. This appears surprising since one may arrive at this rule by making the assumption of conditional independence of the input features given the output class [14]. This assumption is certainly not true in general – neither different feature representations derived from nor different spectral subbands of the same signal are conditionally independent given the class [25]. On the other hand, producing low-entropy distributions over output classes from a product of (sometimes incorrect) classifiers might outweigh this inaccuracy. Alternatively, as argued in Bilmes [26], an assumption that is incorrect for predictive accuracy does not ensure discriminative inaccuracy.

In previous work [23], we additionally investigated other rules such as the

max rule:

$$P(c|\mathbf{x}_1, \ldots, \mathbf{x}_N) = \frac{max_n P(c|\mathbf{x}_n)}{\sum_{c=1}^{K} \max_n P(c|\mathbf{x}_n)} \tag{5}$$

and the *min* rule:

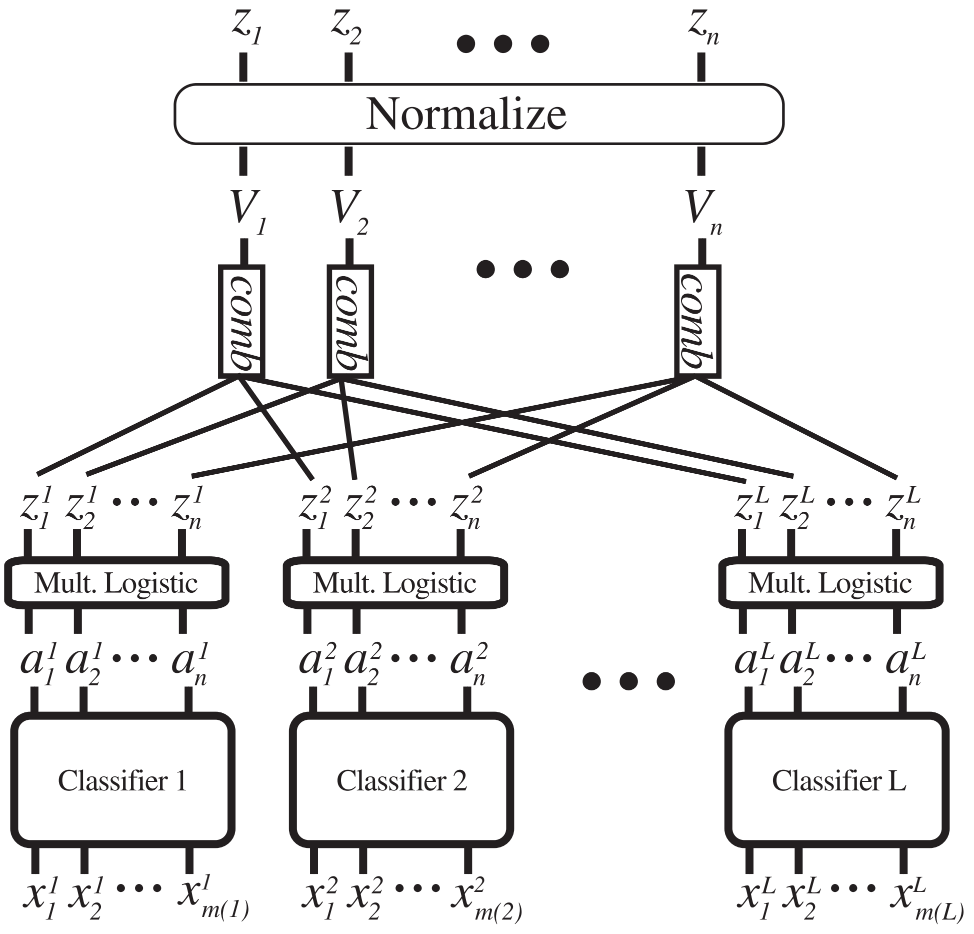
z_1
z_2
z_n
Normalize
V_1
V_2
V_n
comb
comb
comb
$z^1_1\ z^1_2 \cdots z^1_n$
$z^2_1\ z^2_2 \cdots z^2_n$
$z^L_1\ z^L_2 \cdots z^L_n$
Mult. Logistic
Mult. Logistic
Mult. Logistic
$a^1_1\ a^1_2 \cdots a^1_n$
$a^2_1\ a^2_2 \cdots a^2_n$
$a^L_1\ a^L_2 \cdots a^L_n$
Classifier 1
Classifier 2
Classifier L
$x^1_1\ x^1_2 \cdots x^1_{m(1)}$
$x^2_1\ x^2_2 \cdots x^2_{m(2)}$
$x^L_1\ x^L_2 \cdots x^L_{m(L)}$

Fig. 1 Architecture and notational definitions used in this paper. Input features x^l are presented to the lth classifier. These produce linear outputs a^l which are then normalised using the multiple logistic function. For each output k of each of the L classifiers, the values are combined using one of the combination methods, and this produces value V_k. The result is once again normalised producing the final probability mass function z at the output of the combined system

$$P(c|\mathbf{x}_1, \ldots, \mathbf{x}_N) = \frac{min_n P(c|\mathbf{x}_n)}{\sum_{c=1}^{K} \min_n P(c|\mathbf{x}_n)} \tag{6}$$

We found that significant error reductions occurred with the AND-style rules (product and *min*), whereas the sum and the *max* rule (OR style rules) yielded only slight improvements and sometimes even worsened global performance.

In the following sections, we present new AND-style combination rules which generalize the standard combination rules, and can also be used as joint classifier training schemes.

Basic combination architecture

This section describes our combination architecture. We use L neural network classifiers, each of which is a multi-layer perceptron (MLP). Each classifier uses the multiple logistic[2] nonlinearity in the final layer:

$$z^l_k = \frac{exp(a^l_k)}{\sum_{k'} exp(a^l_{k'})} \tag{7}$$

where a^l_k is the kth linear output of the lth classifier *before* the multiple logistic function is applied, and z^l_k is the kth output of the lth classifier *after* applying the multiple logistic function. The latter are combined using one of the many possible and soon-to-be-defined combination rules:

$$V_k = \text{combination\_rule}(z^1_k, z^2_k, \ldots, z^L_k) \tag{8}$$

The combined outputs are re-normalised, thereby producing the final combined system outputs

$$z_k = \frac{V_k}{\sum_j V_j}. \tag{9}$$

This architecture is depicted in Fig. 1.

Under normal circumstances, each classifier is trained separately using the standard back-propagation algorithm. During testing the outputs of each of the sub-classifiers are combined using a combination rule, and then used in subsequent stages of classification. In Sect. 5 we consider methods to jointly train the different classifier systems.

[2] Often referred to as the 'softmax' function, but we call it the *multiple logistic function* in this paper to avoid any confusion with our soft minimum and maximum functions defined in Sect. 4.

Generalized combination rules

This section presents new 'soft' continuous rules that generalise the often used 'hard' combination rules such as the mean (Eq. (1)), product (Eq. (4)), *min* (Eq. (6)), *max* (Eq. (5)). This is done by defining a variety of *soft minimum* functions, and then by showing how they generalise other rules. Each function is dependent on a softness parameter β: the larger the absolute value of β, the harder the function becomes. For notational convenience, we also define single letter versions of the function using a superscripted V.

We define the *smin* function as follows:

$$V_k^{(s)} \stackrel{\triangle}{=} smin_\beta(z_k^1, z_k^2, \ldots, z_k^L) \stackrel{\triangle}{=} \left(\sum_{l=1}^{L} (z_k^l)^{-\beta} \right)^{-1/\beta} \tag{10}$$

A second rule useful when the arguments are probabilities ($0 < z_k < 1\ \forall k$) is defined as follows:

$$V_k^{(p)} \stackrel{\triangle}{=} psmin_\beta\,(z_k^1, z_k^2, \ldots, z_k^L)$$
$$\stackrel{\triangle}{=} exp \left(- \left(\sum_{l=1}^{L} (\ln 1/z_k^l)^\beta \right)^{1/\beta} \right)$$

Note that $psmin_\beta\,(z_k^1, z_k^2, \ldots, z_k^L) = \exp(-smin_{-\beta}\,(-\ln(z_k^1), -\ln\,(z_k^2), \ldots, -\ln\,(z_k^L)))$.. We define a third min function as follows:

$$V_k^{(e)} \stackrel{\triangle}{=} esmin_\beta\,(z_k^1, z_k^2, \ldots, z_k^L) = \sum_{l=1}^{L} \frac{z_k^l e^{-\beta z_k^l}}{\sum_{l=1}^{L} e^{-\beta z_k^l}} \tag{11}$$

as well as a corresponding version when the input lies within the range $0 < z_k < 1$:

$$V_k^{(q)} \stackrel{\triangle}{=} qsmin_\beta\,(z_k^1, z_k^2, \ldots, z_k^L) \tag{12}$$

$$= \exp \left(\sum_{l=1}^{L} \frac{\ln\,(z_k^l)\,(1/z_k^l)^\beta}{\sum_{l=1}^{L} (1/z_k^l)^\beta} \right) \tag{13}$$

Similar to the above, we have that

$$qsmin_\beta\,(z_k^1, z_k^2, \ldots, z_k^L) = \exp\,(-\ esmin_{-\beta}\,(-\ln\,(z_k^1),\ -\ln\,(z_k^2), \ldots, -\ln\,(z_k^L))) \tag{14}$$

For all of the above soft minimum functions, there exists a dual 'soft maximum' function[3] obtained by negating the value of β. For example, we may define a function $smax_\beta\,(\cdot) \stackrel{\triangle}{=} smin_{(-\beta)}\,(\cdot)$. Note that all the minimum functions approach the true *min* function as β gets large since for each soft min function:

$$\min\,(z_1, z_2, \ldots, z_L) = \lim_{\beta\to\infty} {*}min_\beta\,(z_1, z_2, \ldots, z_L) \tag{15}$$

These soft functions are useful because they approximate the minimum (resp. maximum) functions as β gets large and positive (resp. as β gets large and negative). These functions are also continuous and differentiable with respect to their arguments. And surprisingly, they generalise most of the functions that are commonly used in classifier combination systems. For example, $smin_{-1}$ is the sum rule, $smin_1$ is a scaled harmonic mean, $psmin_1$ is the product rule, $esmin_0$ is the mean, and so on. For certain values of β and certain combination methods, some interesting new rules result, such as a 'harmonic product' rule using *psmin* with $\beta = -1$. Figure 2 depicts all the generalisations made by the various soft combination rules for different β values.

[3] Not to be confused with the standard softmax function used for neural networks which, in this paper, we refer to as the multiple logistic function.

Derivation of joint training algorithms

As mentioned above, unlike hard combination rules such as *min* and *max*, the soft versions are continuous and differentiable. Therefore, a new learning algorithm can be defined that jointly trains the L networks. According to the analysis presented in Sect. 2, a joint training rule should encourage the classifiers to perform as well as possible, but should also encourage any errors, if they must be made, to be as statistically independent as possible.

It might at first seem counter-intuitive to jointly train networks in an attempt to produce independent errors. With separate training, however, there is no independence guarantee, instead there is only the hope that the solution arrived at by each classifier will have this property. On the other hand, by using an appropriate joint training rule, the error dependence may be adjusted in a controlled fashion, encouraging the classifiers to arrive at different solutions when it is advantageous to do so. The well-known boosting technique, for example (described in

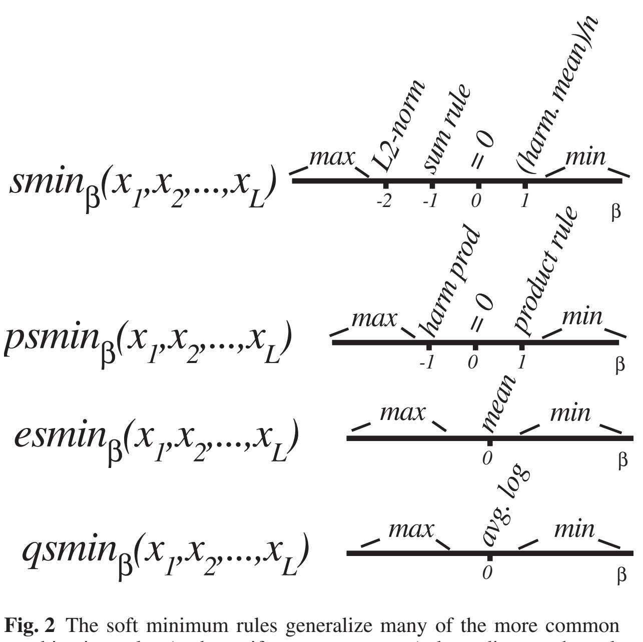

Fig. 2 The soft minimum rules generalize many of the more common combination rules (and specify some new ones) depending on the value of β

Sect. 7) creates multiple classifiers in which later classifiers are trained using information gleaned from earlier ones. A joint training algorithm can be defined for any of our soft combination rules mentioned in Sect. 4. The following analysis will be needed.

First, we need the derivatives of the soft minimum functions. The first one is as follows:[4]

$$\frac{\partial V_k^{(s)}}{\partial z_j^l} = \left(\frac{V_j}{z_j^l}\right)^{1+\beta} \delta_{jk} \tag{16}$$

Note that when $\beta > 1$, the derivative with respect to the smallest argument of *smin* has the largest value. When β gets large, this gradient approaches unity, while the derivative with respect to larger arguments approaches zero. The exact opposite is true when $\beta < 1$ and gets smaller, i.e. the derivative with respect to the largest element has the largest value. This behaviour is as expected, since for the *smin* function, the outputs of the networks other than the minimum do not survive combination when β is large enough. Therefore, they need not change. Only the network with the smallest output should have a non-zero gradient.

The derivatives for the other three soft-min rules are as follows:

$$\frac{\partial V_k^{(e)}}{\partial z_j^l} = \frac{V_j}{z_j^l} (\ln z_j^l / \ln V_j)^{\beta-1}\, \delta_{jk} \tag{17}$$

$$\frac{\partial V_k^{(p)}}{\partial z_j^l} = \frac{e^{-\beta z_j^l}}{\sum_l e^{-\beta z_j^l}} (1 - \beta z_j^l + \beta V_j)\delta_{jk} \tag{18}$$

and

$$\frac{\partial V_k^{(q)}}{\partial z_j^l} = \frac{V_j}{(z_j^l)^{\beta+1} \sum_l (z_j^l)^{-\beta}} (1 - \beta \ln z_j^l + \beta \ln V_j)\delta_{jk} \tag{19}$$

These derivatives have interpretations similar to Eq. (16), i.e. when β gets large, the derivatives approach unity only when j corresponds to the index for the smallest argument.

For a cost function, we use the relative entropy between the targets t_k and the final combined network outputs z_k (i.e. $J = \Sigma_k \ln t_k \ln t_k / z_k$). As in normal back-propagation [7], we compute $\frac{\partial J}{\partial w}$ for each weight w in all of the networks, and perform gradient descent. When the classifiers are MLP-based, the difference between independent training and joint training is that each network has an output-layer 'delta' [7] dependent on the other networks (the hidden-layer deltas and the remaining update steps are identical). This can be seen by noting that

$$\frac{\partial J}{\partial z_j^l} = \sum_k \frac{\partial J}{\partial z_k} \frac{\partial z_k}{\partial z_j^l} = \frac{1}{V_j} \frac{\partial V_j}{\partial z_j^l} (z_j - t_j) \tag{20}$$

[4] Here, δ_{ij} is the Dirac delta.

which leads to the definition of the output delta for the kth output of the lth network:

$$\delta_k^l = \frac{\partial J}{\partial a_j^l} = \sum_j \frac{\partial z_j^l}{\partial a_k^l} \frac{\partial J}{\partial z_j^l}$$

$$= \sum_j (\delta_{jk} - z_k^l) \frac{z_j^l}{V_j} \frac{\partial V_j}{\partial z_j^l} (z_j - t_j)$$

In this form, the derivative of the appropriate soft minimum rule (or in fact, any rule possessing a derivative) may be substituted in place of $(\partial V_j / \partial z_j^l)$ in the above to obtain the final output layer deltas. First, define $\alpha_{jk} \triangleq (\delta_{jk} - z_k^l)(z_j - t_j)$. For each of the soft minimum functions, we get the following output layer deltas:

$$\delta_k^{l(s)} = \sum_j \alpha_{jk} \left(\frac{V_j}{z_j^l}\right)^{\beta} \tag{21}$$

$$\delta_k^{l(p)} = \sum_j \alpha_{jk} (\ln z_j^l / \ln V_j)^{\beta-1} \tag{22}$$

$$\delta_k^{l(e)} = \sum_j \alpha_{jk} \frac{z_j^l e^{-\beta z_j^l}}{\sum_l z_j^l e^{-\beta z_j^l}} (1 - \beta z_j^l + \beta V_j) \tag{23}$$

$$\delta_k^{l(q)} = \sum_j \alpha_{jk} \frac{1}{(z_j^l)^{\beta} \sum_l (z_j^l)^{-\beta}} (1 - \beta z_j^l + \beta V_j) \tag{24}$$

We provide an intuitive explanation of Eq. (21) which can be expanded as follows:

$$\delta_k^{l(s)} = \left(\frac{V_k}{z_k^l}\right)^{\beta} (z_k - t_k) - z_k^l \sum_j (z_j - t_j) \left(\frac{V_j}{z_k^l}\right)^{\beta} \tag{25}$$

This equation says that $\delta_k^{l(s)}$ is the difference of two terms. First, if there is a single network ($L = 1$), one obtains the normal delta for relative entropy, $z_k - t_k$. In the general case where $L > 1$, the first term is similar to the normal delta term for single network training (weights are decreased if $z_k > t_k$) except that the difference is weighted according to how close the output z_k^l is to the determining element. The determining element, in this case, is defined as the minimum (respectively, maximum) if $\beta > 1$ (respectively, if $\beta < 1$). The second term measures how well the combined classifier system is doing overall on this training pattern, but this score is an average over all output units, each weighted according to the proximity of that unit to the determining element.

Note that with this rule delta, if one network classifier is right and the other is in error, both networks will be encouraged to produce the right solution. However, if $\beta > 0$, then the network that was in error will be allowed to pursue a more relaxed solution (i.e. other outputs will be encouraged to increase) because during the minimum-like combination, those additions will not influence the final result. This delta rule then, in some sense, corrects both networks in response to errors, but allows (and perhaps encourages) them to come up with quite different solutions to the problem.

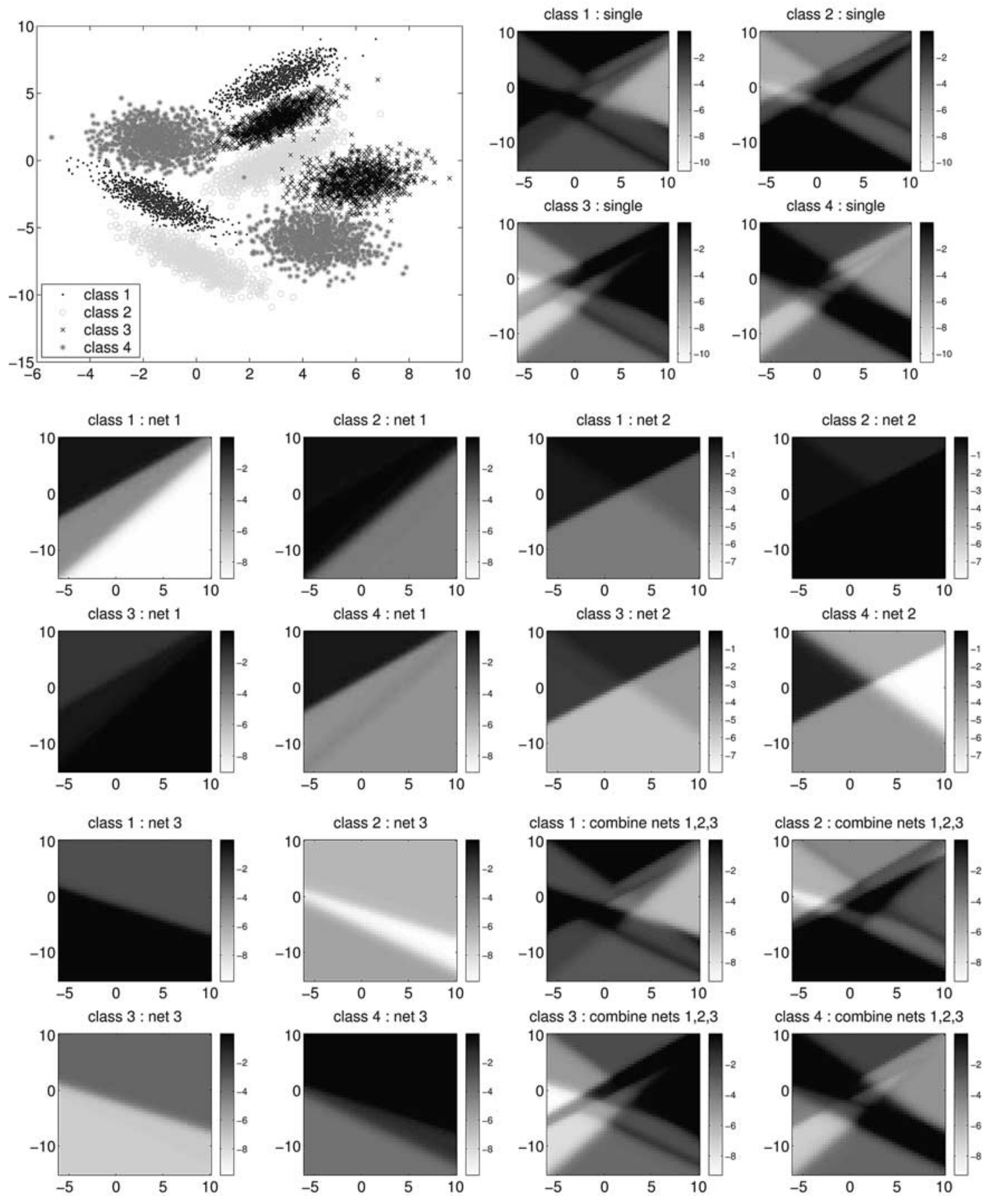

Fig. 3 Example classifier combination using psmin rule and $\beta = 1$. Top left: 8000 samples from four classes. Top right: The response regions for a single network classifier. Middle left and right, and bottom left: The response regions for single network classifiers used in the joint classifier. Bottom right: the final joint classifier, combining the previous three

To further understand our combination rules, we explore the case of simple Gaussian-mixture classification data in two dimensions. We compare a single classifier with that of a combination of three classifiers trained and then combined using the *psmin* rule. The results are presented in Fig. 3. The *upper left* figure shows the data, where there are four classes in two-dimensional space, each class consisting of a mixture of two equally weighted Gaussians. Note that each class contains two non-contiguous regions in this space. The *upper right* figure shows the solution obtained by a three-layer MLP with seven hidden units and four outputs (one for each class) trained on this data using the KL-divergence [22] cost function and a multiple logistic output layer. There is one image per output unit, each one showing in 2 dimensions the degree to which that output is active. The shading depicts log (base 10) probability, and darker shading indicates higher probability. As can be seen, the four units carve up the space so as to classify the data seen in the upper left. The *bottom right* plot shows the result of simultaneously training and then combining with three ($L = 3$) networks using the *psmin* rule with $\beta = 1$.[5] Each pixel for a given two-dimensional location shows z_k in Eq. (9), where k is the class number. As can be seen, the response regions for the combined system are similar to the single system (upper right). The *middle left and right*, and the *lower left* plots show the response regions of the three individual networks, each of which has two hidden units. Each pixel shows z_k^l in Eq. (7), where k is the network output and l is the network number. The product of these three response regions (after normalisation as in Eq. (8)) result in the response regions given in the lower right. As mentioned earlier, *psmin* with positive beta is like an AND function, and it can be seen that the lower right takes a form of intersection of its constituent network outputs. The final response, however, is more than just the intersection because of the effect of the final normalisation (Eq. (8)).

The delta rules for the other soft minimum functions and beta values can be understood in an analogous manner.

Experiments and results

In this section, we evaluate the above combination schemes on speech data as follows. We report results for (1) a baseline system using standard combination rules, (2) the new combination rules applied to independently trained networks, and (3) jointly trained networks.

We use the OGI Numbers95 telephone-speech continuous numbers corpus [21] with 3233 utterances (610,148 frames) for training, 357 utterances (65,029 frames) for cross-validation, and 1206 (226,166 frames) for testing.

The classifiers are trained on manual annotations of phone classes at the frame level; each speech frame in the training set serves as a training sample. We use an ensemble of two ($L = 2$) three-layer MLP classifiers each of which is trained on a different feature representations of the speech wave-form, viz. 'RASTA-PLP' features [27] and Mel-Frequency Cepstral Coefficients (MFCCs). The dimensionality of the RASTA-PLP feature space is 18, that of the MFCC feature space is 26. We use a window of nine frames at the input level, which corresponds to approximately 115 ms of speech. The number of input units in each network is 234 (26×9) and 162 (18×9), respectively. The number of hidden units in the MFCC network is 400; this number was optimised in various development experiments. To compensate for the different dimensionalities of the feature spaces the RASTA-PLP network has 578 hidden units. The number of output units in both networks is 32 (which equals the number of phones in the system).

Table 1 Results (in % word error rate) for baseline combination experiments using the product, sum, *min* and *max* rules

	MFCC	RASTA	min	max	prod	sum
WER	7.0%	8.4%	6.0%	6.7%	5.9%	7.1%

Table 2 Word error rates for combination of *individually* trained classifiers with different soft *min* rules

	smin	psmin	esmin	qsmin
WER	5.1%	5.1%	5.3%	5.2%

In the baseline system, the MLPs are independently trained using back-propagation, a KL-divergence [22] based cost function, and the multiple logistic function in the output layer. We trained two bootstrap networks and combined their outputs using the four combination rules (*max*, *min*, product and sum) described above before passing the acoustic classifier scores to the search component. The results are shown in Table 1.

Similar to our previous studies [18,23], we found that the product rule achieves the lowest word error rate.[6] After several more (using only product combination of acoustic scores) training iterations, the best baseline system achieved a minimum word error rate of 5.1%.

We then evaluated the combination of individually trained classifiers using the new combination rules with a variety of β values in the range -20 to $+20$ (note again that negative β values result in soft maximum rules). Results are reported for the best-performing cases (see Table 2). The β values for these cases ranged from 1–10, so that all best-performing combination schemes are *min*-like rules. We can see, however, that there is no improvement over the result obtained by the standard product rule mentioned above.

In our next set of experiments, we jointly trained and combined the acoustic classifiers using the soft combination schemes. The results are shown in Table 3. We notice a marked drop in word error rate for the *psmin* rule – the β values in this case was 1, so that this combination scheme is equivalent to the product rule, as mentioned above (see Fig. 2). This reduction in word error rate is significant with ($p < 0.002$) and was stable across a range of experimental conditions.[7]

[5] This therefore corresponds to the product rule. We use this rule and beta value because of visual simplicity. In this simple case, the error performance of each of the systems is comparable.

[6] Word error rate is a standard evaluation measure in speech recognition and is defined as $100 \times (1 - (N - I - D - S)/N)$, where N is the number of words, I is the number of word insertions, D is the number of word deletions, and S is the the number of word substitutions, all determined according to a dynamic programming alignment between the recogniser output and the correct transcription.

[7] Specifically, conditions where a varying exponential weight factor was applied to the language model in order to simulate various degrees of acoustic reliability. This is typically called a language model weight factor in the speech recognition field.

Table 3 Word error rates for combination of *jointly* trained classifiers with different soft min rules

	smin	psmin	esmin	qsmin
WER	5.2%	4.8%	5.2%	5.3%

Table 4 Correlation coefficients for classifier outputs under individual and joint training schemes

Training scheme	smin	psmin	esmin	qsmin
Individual training	0.71	0.68	0.70	0.70
Joint training	0.31	0.07	0.59	0.31

Table 5 Phone classification accuracy rates under individual and joint training schemes

Training scheme	smin	psmin	esmin	qsmin
Individual training	80.13%	80.65%	79.06%	80.07%
Joint training	79.55%	78.22%	78.32%	79.53%

The most natural explanation for this drop in word error rate would be to assume that the joint training process reduces the correlation among the networks' errors, which improves phone classification accuracy, and thereby word error rate. To verify this assumption we computed the error correlation on the outputs of the individually-trained networks and the jointly-trained networks, as well as the phone classification accuracies.

The correlation coefficients (see Table 4) indeed show that error correlation is much lower for the jointly trained classifiers. However, phone classification (Table 5) does not improve but even decreases through joint training. The observation that improvements in phone classification accuracy does not necessarily lead to a lower word error rate has in fact often been made in the speech recognition community.

The reason why lower error correlation leads to an improvement in word recognition stem from the interactions between the phone probability distributions generated by joint training/combination and the higher-level search component. A cursory analysis of the average entropy values of the phone probability distributions shows that a reduction in word error rate is often achieved when the average entropy for correctly classified speech frames is low and the average entropy for incorrectly classified frames is relatively high. When the frames are correctly classified and the distributions have low entropy, the search component is less likely to confuse correct states with incorrect states (since the correct state gets almost all of the probability mass). When the frames are incorrectly classified and the distributions have high entropy, these local incorrect phone classification decisions do not result in the correct path being pruned (or significantly reduced in probability) from the search space. This is because the set of alternative states receive probabilities close to that of the top-scoring state.

In our system, both individual classifiers already achieve reasonable phone classification accuracy (77.18% for the MFCC-based classifier, 78.74% for the RASTA-based classifier), so much of the time both classifiers will produce the correct result. The low error correlation, however, implies that a product-like combination should improve our results. Specifically, combination using the product rule will reduce the entropy of the combined output distribution when the individual classifiers agree. Consider, for example, a ternary distribution with probability values $p_1 = 0.1$, $p_2 = 0.8$, $p_3 = 0.1$ which has entropy $H = 0.92$. The normalised product of this distribution with itself will have the much lower entropy $H = 0.23$. On the other hand, in those cases where systems make errors and assign most of the probability mass to *different* phone classes, the entropy of the product distribution will increase compared to those of the individual classifiers. Consider, for example, two ternary distributions p^A and p^B with values $p_1^A = 0.1, p_2^A = 0.8, p_3^A = 0.1$ and $p_1^B = 0.8, p_2^B = 0.1, p_3^B = 0.1$, each distribution again having entropy $H = 0.92$. In this case, the normalised product results in an increase in entropy, to $H = 1.26$.

This leads to the desired kind of interaction between classifier output and search component described above, or, more generally, for any pattern recognition system which performs search over a state space with the goal of finding the globally optimal path. The classifier training and combination schemes described in this paper might therefore also improve the performance of recognition systems for similar tasks, such as HMM-based handwriting recognition.

Comparison with related work

In this section we compare our approach to other techniques previously proposed in the literature. As mentioned above, a relatively simple way of creating ensembles of dissimilar classifiers (i.e. classifiers whose errors are independent) is to employ different randomly selected training sets, different initial conditions or different types of classifiers. These methods are somewhat *ad hoc* and do not guarantee that the resulting classifiers will in fact differ in their classification errors. More principled ensemble training methods include Bagging and Boosting. Bagging [28] uses different training sets for each classifier in the ensemble. Each training set is generated by randomly re-sampling the original training set. The outputs of the resulting classifiers are then combined by simple averaging. In this method, classifiers are trained independently. Boosting [29,30] also uses a re-sampling technique; here, however, the training samples are chosen based on the performance of previously trained classifiers. The probability of selecting a sample for the next training set is proportional to the number of times it was misclassified by previously trained classifiers. The outputs of all classifiers are combined by weighted voting, where the weights are dependent on the classifiers' accuracies on the training set. In this method, classifiers are trained jointly only in the sense that the latter classifiers are trained based on information obtained from earlier already trained classifiers. The boosting technique has been previously applied to the same task presented in this work. In that case, boosting achieved a word error rate of 5.3% [31]. Our best joint training and combination result presented herein, however, is 4.8% using a system with significantly fewer parameters.

In Rosen [32] an ensemble of neural network classifiers is trained using an objective function which includes a penalty term for correlated network outputs in addition to the normal mean squared error term. During training, the correlation penalty term is minimised. This technique was tested on learning the three-parity problem, a noisy sine function, and a noisy nonlinear function. Performance, measured in terms of mean squared error between the networks' outputs and the targets, was shown to improve when the decorrelation term is included in the objective function. It should be noted that under this approach, networks are also trained sequentially: the modified training criterion is applied after at least one member of the neural network ensemble has been trained using a standard training criterion. The goal is to minimise the correlation between the current network's output and that of previously trained networks. Our technique is different, in that it trains all ensemble members concurrently rather than sequentially.

Another explicit attempt at minimising the error correlation in a neural network ensemble is described in [33]. Here, a genetic algorithm is employed to search the space of possible networks with the objective of finding ensemble member that are both accurate and diverse. From an initial population of networks, new networks are created using genetic operators. Those networks are retained in the population which maximise a combined fitness criterion consisting of an accuracy and a diversity term. This algorithm was tested on the task of gene sequence localisation and outperformed ensembles created by Bagging or randomly choosing network structures. The best performance was obtained by combining genetic search for optimal ensemble members with a training procedure which, at each iteration, emphasised the training samples which were misclassified by the current ensemble.

Summary

In this paper, we have presented new classifier combination techniques that generalise previously used combination rules, such as the mean, product, *min* and *max* functions. These new continuous and differentiable forms can be used both for combination of independently trained classifiers and as objective functions in new joint classifier training schemes. We demonstrated the application of these rules to both combination and joint training of acoustic classifiers in a speech recognition system and analysed their effect on word recognition performance. We found a significant word error rate improvement over previous combination schemes when training and combining classifiers with a version of the product rule. An analysis of this result suggests that the improvement is due to the fact that the joint training scheme produces output probability distributions which more favourably interact with the higher-level recognition component that performs a search for the globally optimal path. In the future, we shall investigate training the β exponent in addition to the network weights and extending these schemes to various other tasks beside speech recognition, such as single level classification.

References

1 Kittler J, Hataf M, Duin RPW and Matas J, On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence 1998; 20(3): 226–239

2 Jacobs RA, Methods for combining experts' probability assessments. Neural Computation 1995; 7: 867–888

3 Krogh A and Vedelsby J, Neural network ensembles, cross validation, and active learning. Advances in Neural Information Processing Systems 7. MIT Press, 1995

4 Perrone MP and Cooper LN, When networks disagree: ensemble methods for hybrid neural networks. In RJ Mammone, editor, Neural networks for speech and image processing. Chapman & Hall, 1993; 126–142

5 Wolpert DH, Stacked generalization. Neural Networks 1992; 5: 241–259

6 Sharkey AJC, Multi-net systems. In Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems. Springer, 1999, pp 3–30

7 Bishop C, Neural networks for pattern recognition. Clarendon Press, Oxford, 1995

8 Tumer K and Ghosh J, Analysis of decision boundaries in linearly combined neural classifier. Pattern Recognition 1996; 29: 341–348

9 Sharkey AJC, Sharkey NE and G Chandroth GO, Neural nets and diversity. In Proceedings 14th international conference on Computer safety, reliability and security, 1995; 375–389

10 Ho T-K, Hull JJ and Srihari SN, Decision combination in multiple classifier systems. IEEE Transactions on Pattern Analysis and Machine Intelligence 1994; 16(1): 66–75

11 Hashem S, Optimal linear combinations of neural networks. Neural Networks 1997; 10(4): 599–614

12 Rogova G, Combining the results of several neural network classifiers. Neural Networks 1994; 7: 777–781

13 Lam L and Suen CY, Optimal combinations of pattern classifiers. Pattern Recognition Letters 1995; 16: 945–954

14 Bilmes JA and Kirchhoff K, Directed graphical models of classifier combination: Application to phone recognition. In Proceedings international conference on spoken language processing, Beijing, China, 2000

15 Bourlard H and Morgan N, Connectionist speech recognition: a hybrid approach. Kluwer Academic, 1994

16 Kingsbury BED and Morgan N, Recognizing reverberant speech with RASTAPLP. In Proceedings ICASSP-97, 1997

17 Halberstadt AK and Glass JR, Heterogeneous measurements and multiple classifiers for speech recognition. In Proceedings international conference on spoken language processing, 1998; 995–998

18 Kirchhoff K, Combining acoustic and articulatory information for speech recognition in noisy and reverberant environments. In Proceedings international Conference on spoken language processing, 1998

19 Mirghafori N and Morgan N, Combining connectionist multi-band and full-band probability streams for speech recognition of natural numbers. In Proceedings international conference on spoken language processing, 1998; 743–746

20 McMahon P, Court P and Vaseghi S, Discriminative weighting of multiresolution sub-band cepstral features for speech recognition. In Proceedings international conference on spoken language processing, 1998; 1055–1058

21 Cole RA, Noel M, Lander T, Durham T, New telephone speech corpora at CLSU. In European conference on speech communication and technology (Eurospeech), 1995; 821–824

22 Cover TM and Thomas JA, Elements of information theory. Wiley, 1991

23 Kirchhoff K and Bilmes J, Dynamic classifier combination in hybrid

speech recognition systems using utterance-level confidence values. Proceedings international conference on acoustic, speech and signal processing, 1999; 693–696
24 Hinton GE, Training products of experts by minimizing contrastive divergence. Technical report, Gatsby Computational Neuroscience Unit, 2000
25 Bilmes J, Natural statistic models for automatic speech recognition. PhD thesis, UC Berkeley, 1999
26 Bilmes JA, Dynamic Bayesian Multinets. In Proceedings 16th conference on uncertainty in artificial intelligence. Morgan Kaufmann, 2000
27 Hermansky H and Morgan N, RASTA processing of speech. IEEE Transactions on Speech and Audio Processing 1994; 2(4): 578–589
28 Breiman L, Bagging predictors. Machine Learning 1996; 26: 123–140
29 Schapire RE, The strength of weak learnability. Machine Learning 1990; 5: 197–227
30 Freund Y and Schapire R, Experiments with a new boosting algorithm. In Proceedings 13th international conference on machine learning 1996; 148–156
31 Schwenk H, Using boosting to improve a hybrid HMM/neural network speech recognizer. In International conference on acoustics, speech and signal processing, Phoenix, AZ, 1999; 1009–1012
32 Rosen B, Ensemble learning using decorrelated neural networks. Connection Science (Special Issue on Combining Artificial Neural Nets: Ensemble Approaches) 1996; 8(3–4): 373–384
33 Opitz D and Maclin R, Popular ensemble methods: an empirical study. Journal of Artificial Intelligence Research 1999; 11: 169–198

Jeff A. Bilmes is an Assistant Professor at the Department of Electrical Engineering at the University of Washington, Seattle, and is also an adjunct Assistant Professor in Linguistics. He co-founded the Signal, Speech, and Language Interpretation Laboratory at the University. He received a masters degree from MIT, and a PhD in Computer Science at the University of California in Berkeley in 1999. Jeff was also a leader of the 2001 Johns Hopkins summer workshop on speech recognition. His primary research interests lie in statistical modelling (particularly graphical model approaches) and signal processing for pattern classification, speech recognition, language processing, and audio processing. He also has strong interests in information theory, speech-based human-computer interfaces, high performance parallel computing systems, computer architecture, and software optimisation techniques. He is an associate editor of the *IEEE Transactions on Multimedia*, is a 2001 recipient of the NSF CAREER award, and is a 2001 CRA Digital-Government Research Fellow.

Katrin Kirchhoff holds an MA degree in English Linguistics (1996) and a PhD in Computer Science (1999) from the University of Bielefeld, Germany. She is currently a Research Professor in the Department of Electrical Engineering at the University of Washington, Seattle. Her research interests include machine learning, automatic speech recognition, and natural language processing.

Originality and Contribution

Linear combination rules such as the product, sum, *min* or *max* rule are well-known and widely used in the classifier combination community. In this paper, we present a set of four new combination rules which can be interpreted as soft versions of a *min* (or *max*) rule. Depending on the value of the ‘softness’ parameter, these rules also generalise the other aforementioned rules, as well as a range of other functions. The new combination rules have a continuous and differentiable form and can therefore be used not only for the combination of individually trained classifiers, but they can also be employed as objective functions for jointly training classifiers. In this paper we provide both the rules and their derivatives needed for a joint training scheme. To our knowledge, an analysis of these or similar trainable rules for combining probability distributions has not been published before. The mathematical background provided in this paper is sufficient to enable other researchers to use these combination schemes for any pattern classification task of their choice.

We evaluate these rules by applying them to phone classifiers in a speech recognition system. Our results show a significant improvement in word recognition performance when using classifiers trained jointly with a soft version of the product rule. Analysis of these results indicates that the gain in performance stems from the improved interaction of sequential classification components in the recognition system due to the joint training approach. A further contribution of this paper is the analysis of optimal conditions for classifier combination in a multi-level as opposed to a single-level classification system.