TMR PHYSTA Report [1]
Confidence estimation and on line retraining of neural networks

S Stroeve, S Gielen
University of Nijmegen, The Netherlands

A Doulamis, N Doulamis, S Kollias
National Technical University of Athens, Greece

R Cowie, E Cowie-Douglas, S McGilloway
Queens University of Belfast, United Kingdom

September 15, 1999

[1]'Principled Hybrid systems: Theory and Applications'
Research contract FMRX-CT97-0098 (DG 12 - BDCN)

Contents

1 Introduction

Neural networks are applied most commonly in regression and classification tasks. Usually within such tasks, neural networks estimate the first-order statistics of the problem at hand, such as mean output, most probable output, etc. Additional estimation of the second-order statistics facilitates the formulation of error bounds for the regression or classification task, which specify the confidence in the network predictions. In the context of the PHYSTA project these error bounds are mandatory for combining the predictions of different networks in a hierarchical hybrid architecture. Furthermore, confidence levels may guide retraining procedures.

In this report we provide some background on error and confidence bounds applied in general statistics and give an overview on the application of such techniques in neural networks. Section 2 discusses sources of uncertainty and some basic principles of error bounds. Section 3 discusses error bounds in non-parametric inference using the bootstrap method. Section 4 deals with error bounds for parameteric maximum likelihood inference. Section 5 shows how the bootstrap method may be applied to ensembles of neural networks to obtain error estimates. Section 6 discusses parametric inference methods for neural networks in combination with a Bayesian perspective on maximum likelihood learning. Low confidence in the network predictions may be the result of nonstationarity of the application. Section 7 discusses a retraining methodology which can detect such nonstationary behaviour and adapt the network accordingly to its new environment. In Section 8 some of these methods are applied on confidence estimation of the emotion classification based on speech features.

2 Standard errors, confidence and prediction intervals

Suppose we are given a set D of N pairs $\{\mathbf{x}^n, t^n\}$ which are generated by a deterministic function f and zero-mean additive noise ξ:

$$t(\mathbf{x}) = f(\mathbf{x}) + \xi(\mathbf{x}) \tag{1}$$

We want to predict function values $f(\mathbf{x})$ using a system $y(\mathbf{x};\mathbf{w})$ with parameters $\mathbf{w}$. The parameters $\mathbf{w}$ are optimized for the set D such that $y(\mathbf{x};\mathbf{w})$ is an estimate of the mean of the target distribution t($\mathbf{x}$), and thus of f($\mathbf{x}$) given the zero-mean noise. The accuracy of this estimate is specified by *confidence intervals*, related to the distribution $f(\mathbf{x}) - y(\mathbf{x};\mathbf{w})$, if we are interested in its relation to $f(\mathbf{x})$. For predictions it is mandatory to account for the noise ξ, such that the relevant probability distribution is $t(\mathbf{x}) - y(\mathbf{x};\mathbf{w})$. Now the accuracy of the predictions is given by the *prediction intervals* related to this distribution. Clearly, prediction intervals enclose confidence intervals, since they take account of the noise ξ in addition to the confidence distribution.

Knowledge of the actual distribution $f - y$ would straightforwardly lead to calculation of the confidence levels. Efron and Tibshirani (1993) discuss bootstrap methods to estimate probability distributions (histograms), but these methods require large computational resources and do not seem to be tractable (yet) for neural network predictions. Therefore, in general, it is assumed that $f - y$ has a Gaussian distribution. Now only an estimate of the standard deviation $\hat{\sigma}$ is required to come up with a confidence interval. If $z^{(\alpha)}$ denotes the 100.αth percentile point of a zero-mean normal distribution with unity standard deviation, then

$$p(z^{(\alpha)} \leq (f - y)/\hat{\sigma} \leq z^{(1-\alpha)}) = 1 - 2\alpha \tag{2}$$

and the standard confidence interval with confidence level $100(1-2\alpha)\%$ is $y \pm z^{(1-\alpha)}\hat{\sigma}$.

Assuming a normally distributed noise process ξ, prediction intervals can additionally be determined using estimates of its variance.

The following potential sources of uncertainty for the network predictions may give rise to the confidence intervals:

Finite data set The data set D contains a finite number of samples N taken from the target probability distribution. This restricts the accuracy of the estimates.

Incomplete data It might be that for a particular task we are not aware of all the relevant input variables. Thus, if $\mathbf{x}$ is the relevant input vector, only a subset $\mathbf{x_1}$ of the input variables is available for the regression $y(\mathbf{x_1};\mathbf{w})$. Such an error in the model structure may give rise to large variations in the target variables, which can not be explained on the basis of the input signals.

Missing values In the case that we do know the relevant input variables, some of them may be missing in the data set D. If N is not so large that we can disregard such incomplete samples, we have to use the incomplete data samples and thus introduce further uncertainty.

Training randomness The final weight configuration of the neural network depends on the training data, the learning rule, as well as the initial random weights. The random initial weight configuration and other random effects of the training procedure thus introduce variance in the network predictions.

3 Bootstrap estimates of standard errors and confidence intervals

Given a sample $\mathbf{x}$ from a random variable x drawn from a probability distribution F, it is easy to attain an error estimate for the mean μ_F. An estimate of the mean and the standard error for the sample $\mathbf{x}$ is attained by using the empirical probability distribution $\hat{F}$, which consists of a series of delta peaks for the data-points. Now estimates for the mean and the standard error of the mean are:

$$\hat{\mu}_F = \mu_{\hat{F}} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{3}$$

$$\hat{\sigma}(\hat{\mu}_F) = \sigma_{\hat{F}}/\sqrt{N} = \sqrt{\frac{1}{N(N-1)}\sum_{i=1}^{N}(x_i-\hat{\mu}_F)^2} \tag{4}$$

For a general statistic other than the mean no straightforward formulas exist for estimates of the standard error. Suppose we want to calculate a parameter $\theta = g(F)$ based on $\mathbf{x}$ and estimate it using a suitable function $\hat{\theta} = h(\mathbf{x})$. What is the standard error of $\hat{\theta}$? The bootstrap method provides an answer (Efron and Tibshirani, 1993). If $\hat{F}$ is the empirical distribution of size N then a bootstrap sample B_i is a realization of $\hat{F}$ with N elements or, stated differently, it is a random collection drawn with replacement from $\hat{F}$. The bootstrap estimate of θ using M bootstrap realizations is

$$\hat{\theta}_B = \frac{1}{M}\sum_{i=1}^{M}\hat{\theta}_{B_i} \tag{5}$$

with $\hat{\theta}_{B_i}$ the estimate based on the empirical distribution B_i, and the estimated standard error is

$$\hat{\sigma}_B = \sqrt{\frac{1}{M-1}\sum_{i=1}^{M}(\hat{\theta}_{B_i} - \hat{\theta}_B)^2} \tag{6}$$

For large M the bootstrap estimate of the standard error equals the standard error based on the emperical distribution $\hat{F}$:

$$\lim_{M\to\infty} \hat{\sigma}_B = \sigma_{\hat{F}} \tag{7}$$

In practice Efron and Tibshirani (1993) found that about 25 to 50 bootstrap replications usually suffice to acquire good estimates of the standard deviation.

As stated in Section 2 the estimated standard deviation can be used to calculate confidence intervals assuming a Gaussian distribution. In addition, the bootstrap method can provide improved confidence measures which do not use the Gaussian assumption or any other model of the underlying probability distribution. By gathering a large number (in the order of 1000) of bootstrap replications it is possible to compose a histogram for $\hat{\theta}$. Based on this histogram a confidence interval can now be calculated. This confidence interval may not be symmetric and thus can account for the skewness of the distribution.

4 Standard error estimation for parametric inference

In parametric inference we assume a model for the data-generating process and optimize the parameters of the model such that it best describes the data at hand. If $p_\theta(x)$ is the assumed probability distribution with parameters θ, then maximum likelihood inference determines an estimate $\hat{\theta}$ such that the likelihood function

$$L(\theta;\mathbf{x}) = \prod_{i=1}^{N} p_\theta(x_i) \tag{8}$$

is maximized:

$$\hat{\theta} = \text{argmax}_\theta L(\theta;\mathbf{x}) \tag{9}$$

In the discrete case $L(\theta;\mathbf{x})$ is the probability of observing the data $\mathbf{x}$. In the continuous case $L(\theta;\mathbf{x})\Delta$ is approximately the probability of observing the data in the small interval $[\mathbf{x};\mathbf{x}+\Delta]$. Taking the logarithm of $L(\theta;\mathbf{x})$ gives us the log-likelihood

$$l(\theta;\mathbf{x}) = \sum_i l(\theta;x_i) = \sum_i \log p_\theta(x_i) \tag{10}$$

The maximum likelihood estimate $\hat{\theta}$ takes its extreme if

$$\frac{\partial l(\hat{\theta},\mathbf{x})}{\partial\hat{\theta}} = \sum_i \frac{\partial l(\hat{\theta},x_i)}{\partial\hat{\theta}} = 0 \tag{11}$$

The Fisher information matrix $\mathbf{H}(\hat{\theta};\mathbf{x})$ represents the rate of change of the likelihood:

$$\mathbf{H}(\hat{\theta}) = -\sum_i \frac{\partial^2 l(\hat{\theta};x_i)}{\partial\hat{\theta}^2} \tag{12}$$

and is inversely related with the confidence associated with the maximum likelihood solution. The maximum likelihood estimator is asymptotically unbiased, i.e. $E(\hat{\theta}) = \theta$. Moreover, it is asymptotically efficient, i.e. it tends to the lower (Cramer-Rao) bound of the variance for the estimation of the mean, and the estimate $\hat{\theta}$ is asymptotically normally distributed with mean θ and covariance $\mathbf{H}^{-1}$.

Given the regression function $y(\mathbf{x};\mathbf{w})$ with $\mathbf{x}$ input variables and $\mathbf{w}$ regression parameters, an estimate of the standard error of the predictions y for an optimal parameter vector $\hat{\mathbf{w}}$ can be deduced by the delta method (Tibshirani, 1996). The delta method estimate of the variance assumes normally distributed noise and uses a Taylor series approximation to obtain:

$$\hat{\sigma}^2(\mathbf{x}) = \mathbf{g}^T\hat{\mathbf{H}}^{-1}\mathbf{g} \tag{13}$$

with $\mathbf{g} = \nabla_{\hat{\mathbf{w}}} y$ the gradient vector and $\mathbf{H} = \nabla^2_{\hat{\mathbf{w}}} y$ the Hessian. This estimate was also used by Mackay (1996) for the MLP (as will be discussed in Section 6).

5 Bootstrapping in ensembles of neural networks

Consider an ensemble of balanced predictors:

$$\bar{y} = \sum_i \alpha_i y_i(\mathbf{x}) \tag{14}$$

where the weighting factors α_i represent our belief in the predictors and obey the restrictions $\alpha_i \geq 0$ and $\sum \alpha_i = 1$. The ensemble ambiguity is the variance of the weighted ensemble around its mean:

$$\bar{a}(\mathbf{x}) = \sum_i \alpha_i a_i(\mathbf{x}) = \sum_i \alpha_i (y_i(\mathbf{x}) - \bar{y}(\mathbf{x}))^2 \tag{15}$$

If we denote the mean quadratic error of the predictors as $\bar{\epsilon}(\mathbf{x})$ and the quadratic error of the mean prediction as $\mathcal{E}(\mathbf{x})$,

$$\bar{\epsilon}(\mathbf{x}) = \sum_i \alpha_i \epsilon(\mathbf{x}) = \sum_i \alpha_i (f(\mathbf{x}) - y_i(\mathbf{x}))^2 \tag{16}$$

$$\mathcal{E}(\mathbf{x}) = (f(\mathbf{x}) - \bar{y}(\mathbf{x}))^2 \tag{17}$$

then we can write the ensemble generalization error as the difference of the weighted average of the generalization errors of the individual networks and the ensemble ambiguity:

$$\mathcal{E}(\mathbf{x}) = \bar{\epsilon}(\mathbf{x}) - \bar{a}(\mathbf{x}) \tag{18}$$

It follows directly from this equation that the ensemble generalization error $\mathcal{E}(\mathbf{x})$ is always smaller than the average of the ensemble errors $\bar{\epsilon}(\mathbf{x})$. A small ensemble generalization error is attained for ensembles with large ambiguity terms and small individual generalization errors. The ensemble ambiguity $\bar{a}$ can be calculated directly. The individual generalization errors can not be determined since the function $f(\mathbf{x})$ is unknown and need to be estimated somehow.

Here we follow the bootstrapping technique of Heskes (1997) to achieve different training sets for the predictors and to estimate the individual generalization errors.

Given a set D of N input-output patterns $\{\mathbf{x}^n, t^n\}$. A bootstrap sample B_i from the set D is a random collection of N data points drawn with replacement from D. The validation error for the i-th network can be determined on the complement of the bootstrap set B_i (denoted by $\tilde{B}_i$):

$$E_i = \langle (y_i - t)^2 \rangle_{\tilde{B}_i} \tag{19}$$

where $\langle . \rangle_{\tilde{B}_i}$ denotes the average over the probability distribution containing delta peaks on the elements of $\tilde{B}_i$. This error on the complement of the bootstrap set can be used as estimate of the generalization error. An improved estimate of the generalization error was proposed by Heskes (1997) and accounts for the expected validation error. Using the estimates of the generalization errors and the ambiguity terms, minimization of the ensemble generalization error, under the constraint that α can be interpreted as a probability, renders the weighting parameters α_i.

The bootstrap estimate of the variance in the case of equal weighting parameters $\alpha_i = 1/M$ is:

$$\hat{\sigma}^2(\mathbf{x}) = \frac{1}{M-1} \sum_{i=1}^{M} (y_i(\mathbf{x}; \mathbf{w}_i) - \bar{y}(\mathbf{x}))^2 \tag{20}$$

Generalization to the case with different weighting parameters α_i leads to:

$$\hat{\sigma}^2(\mathbf{x}) = \frac{1}{1 - \sum_i \alpha_i^2} \sum_i \alpha_i (y_i(\mathbf{x}; \mathbf{w}_i) - \bar{y}(\mathbf{x}))^2 \tag{21}$$

Although the ensemble prediction was illustrated here for neural network predictors, the same procedure can be used for other prediction systems such as expert systems, decision trees, etc. Breiman (1994) used the discussed procedure with equal weights $\alpha_i = 1/M$ for classification trees and called it 'bagging' (bootstrap aggregating). He showed that improved classification rates and regression accuracy could be obtained, but did not deduce confidence levels.

6 Parametric inference for neural network regressors

6.1 The MLP in a Bayesian context

Given a single output network $y(\mathbf{x}, \mathbf{w})$ with input vector $\mathbf{x}$ and weight vector $\mathbf{w}$. In a Bayesian context (Mackay, 1996) we define a prior distribution of the weights $p(\mathbf{w})$

$$p(\mathbf{w}) = \frac{1}{Z_W(\alpha)} \exp(-\alpha E_W) \tag{22}$$

with $Z_W(\alpha)$ a normalization constant such that $\int p(\mathbf{w}) d\mathbf{w} = 1$. Proper generalization of the network can be achieved by favouring small weights which is operationalized by choosing E_W as:

$$E_W = \frac{1}{2} \|\mathbf{w}\|^2 = \frac{1}{2} \sum_{i=1}^{W} w_i^2 \tag{23}$$

with W the number of weights in the network, leading to $Z_W(\alpha) = (2\pi/\alpha)^{W/2}$ Given a regression problem with a target set $D = \{t^1, t^2, \ldots, t^N\}$, where it is assumed that the target variables t^n

are given by a deterministic function with added Gaussian noise. The probability distribution of the target variables thus is

$$
\begin{aligned}
p(t|\mathbf{x}) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(y(\mathbf{x};\mathbf{w}) - t)^2}{2\sigma^2}) && (24)\\
&\propto \exp(-\frac{\beta}{2}(y(\mathbf{x};\mathbf{w}) - t)^2) && (25)
\end{aligned}
$$

with σ^2 the variance of the noise process and $\beta = 1/\sigma^2$. If the data points are drawn independently from this distribution, the conditional probability distribution $p(D|\mathbf{w})$ is

$$
\begin{aligned}
p(D|\mathbf{w}) &= \prod_{n=1}^{N} p(t^n|\mathbf{x}^n, \mathbf{w}) && (26)\\
&= \frac{1}{Z_D(\beta)} \exp(-\frac{\beta}{2} \sum_{n=1}^{N} \{y(\mathbf{x}^n;\mathbf{w}) - t^n\}^2) && (27)\\
&= \frac{1}{Z_D(\beta)} \exp(-\beta E_D) && (28)
\end{aligned}
$$

with E_D an error function and $Z_D(\beta) = (2\pi/\beta)^{N/2}$ a normalization constant. Now Bayes theorem can be applied to obtain the probability distribution of the weights:

$$
\begin{aligned}
p(\mathbf{w}|D) &= \frac{p(D|\mathbf{w})p(\mathbf{w})}{p(D)} && (29)\\
&= \frac{p(D|\mathbf{w})p(\mathbf{w})}{\int p(D|\mathbf{w})p(\mathbf{w})d\mathbf{w}} && (30)\\
&= \frac{1}{Z_S(\alpha,\beta)} \exp(-\beta E_D - \alpha E_W) = \frac{1}{Z_S(\alpha,\beta)} \exp(-S(\mathbf{w})) && (31)
\end{aligned}
$$

with $Z_S(\alpha, \beta)$ a normalization constant.

The (most probable) weight vector $\mathbf{w}^{mp}$ which maximizes the posterior distribution of Eq. 31 is the weight vector which minimizes $S(\mathbf{w})$. For the Gaussian noise model of the target data and the prior distribution of the weights this comes down to minimization of

$$
S(\mathbf{w}) = \frac{\beta}{2} \sum_{n=1}^{N} \{y(x^n;\mathbf{w}) - t^n\}^2 + \frac{\alpha}{2} \sum_{i=1}^{W} w_i^2 \tag{32}
$$

meaning minimization of the sum-of-squares error function with a weight-decay regularization term.

In general, the normalization constant $Z_S(\alpha, \beta)$ cannot be evaluated analytically and simplifying approximations are needed to evaluate the posterior distribution. To obtain such an approximation the Taylor expansion of $S(\mathbf{w})$ around its minimum at $\mathbf{w}^{mp}$ is truncated for terms higher than second-order:

$$
S(\mathbf{w}) = S(\mathbf{w}^{mp}) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^{mp})^T \mathbf{H}(\mathbf{w} - \mathbf{w}^{mp}) \tag{33}
$$

where the matrix $\mathbf{H}$ is the Hessian of the error function S:

$$
\mathbf{H} = \nabla^2 S^{mp} = \beta \nabla^2 E_D^{mp} + \alpha \mathbf{I} \tag{34}
$$

which holds a contribution from the Fisher information matrix $\nabla^2 E_D^{mp}$ and the prior distribution of the weights. For this approximation, the posterior distribution is

$$p(\mathbf{w}|D) = \frac{1}{Z_S^*} \exp\{-S(\mathbf{w}^{mp} - \frac{1}{2}\Delta\mathbf{w}^T \mathbf{H} \Delta\mathbf{w}\} \tag{35}$$

where the normalization constant is

$$Z_S^* = \exp(-S(\mathbf{w}^{mp}))(2\pi)^{W/2} \det(\mathbf{H})^{-1/2} \tag{36}$$

Confidence level of the MLP Using the Gaussian approximation to the posterior probability distribution and assuming that the network function $y(\mathbf{x}, \mathbf{w})$ is well represented by a linear expansion around $\mathbf{w}^{mp}$

$$y(\mathbf{x}; \mathbf{w}) = y(\mathbf{x}; \mathbf{w}^{mp}) + \mathbf{g}^T \Delta\mathbf{w} \tag{37}$$

with

$$\mathbf{g} = \nabla_{\mathbf{w}} y|_{\mathbf{w}^{mp}} \tag{38}$$

the target distribution is

$$p(t|\mathbf{x}, D) = \frac{1}{\sqrt{2\pi\sigma_t^2}} \exp(-\frac{(t - y^{mp})^2}{2\sigma_t^2}) \tag{39}$$

where

$$\sigma_t^2 = 1/\beta + \mathbf{g}^T \mathbf{H}^{-1} \mathbf{g} \tag{40}$$

The variance of the target distribution thus depends on the variance of the noise of the input data $1/\beta$, the network gradient and the information matrix. Note, that the variance in Eq. 40 is equal to the sum of the variance of the noise of the input data $\sigma^2 = 1/\beta$ and a noise term related to the variability in the weights of the network (see Eq. 13).

6.2 Confidence estimation in a RBF network

A radial basis function (RBF) network with basis functions determined by a mixture density estimation provides confidence intervals in a straightforward manner. The mapping of a RBF network with input $\mathbf{x}$, output $\mathbf{y}$, weights w_{kj} and basis functions ϕ_j is given by

$$y_k(\mathbf{x}) = \sum_{j=1}^{M} w_{kj} \phi_j(\mathbf{x}) + w_{k0} \tag{41}$$

The density of the input data is modeled by a mixture model:

$$p(\mathbf{x}) = \sum_{j=1}^{M} \alpha_j \phi_j(\mathbf{x}) \tag{42}$$

with $\phi_j(\mathbf{x})$ a basis function and α_j a mixing coefficient (Bishop, 1995). The mixing coefficient α_j can be regarded as the prior probability of the data points having been generated by the

j-th component of the mixture. Therefore, the mixing coefficients should satisfy the constraints $\sum \alpha_j = 1$ and $\alpha_j \geq 0$. The basis functions can be regarded as class-conditional densities:

$$\phi_j(\mathbf{x}) = p(\mathbf{x}|j) \tag{43}$$

A common choice for the basis function is a Gaussian distribution:

$$\phi_j(\mathbf{x}) = \frac{1}{(2\pi\sigma_j^2)^{d/2}} \exp(-\frac{\|\mathbf{x} - \mu_j\|^2}{2\sigma_j^2}) \tag{44}$$

with d the dimension of $\mathbf{x}$, and μ_j and σ_j^2 the center and variance of the j-th Gaussian. The mixture model parameters are determined by maximizing the likelihood function $L = \prod_n p(\mathbf{x}^n)$ using a gradient or an expectation-maximization algorithm (Bishop, 1995).

After the parameters of the basis functions have been determined, the weights w_{kj} in the second layer of the RBF network (Eq. 41) can be determined by a supervised learning procedure. Combination of the mixture model of Eq. 42 and the RBF network of Eq. 41 now provides a network prediction together with an estimate of the likelihood of the input vector (Bishop, 1994). A low likelihood of the input vector corresponds with a low confidence in the network prediction.

7 On line retrainable neural networks

7.1 Introduction

While training and using a neural network in many real life cases, the adopted assumption of *stationarity* of the network training data is not valid. Consequently, the training set is not able to represent all possible variations or states of the operational environment to which the network is to be applied. Instead, it would be desirable to have a mechanism, which would provide the network with the capability to automatically test its performance and to be automatically retrained when its performance is not acceptable. The retraining algorithm should update the network weights taking into account both the former network knowledge and the knowledge extracted from the current input data. Adaptive training of neural networks in slowly varying non-stationary processes has been a topic of research within the neural network community in the last few years. Heskes and Kappen (1991) provided a general analysis of learning dynamics in changing environments.

Here a new approach is presented for improving the performance of neural networks when handling non-stationary image and video data, including:

- An automatic decision mechanism, which determines when network retraining should take place.
- An algorithm which retrains the network, adapting its weights to the current environment, by applying a nonlinear programming technique.

A maximum a posteriori (MAP) estimation technique which extracts knowledge from the current input data by modeling the network output as a Markov Random Field (MRF) and by optimally selecting pairs of training inputs and corresponding desired outputs can also be included in the proposed formulation.

7.2 Formulation of the problem

As an example, let $x_{i,j}$ $(i = 0 \dots M_1 - 1\ ,\ j = 0 \dots M_2 - 1)$ denote the image intensity at pixel (i, j). In most image/video coding and analysis applications the image is processed in partitions, or blocks, of 8×8 pixels. For classification purposes it is required to classify each block, or a transformed version of the same size, to one of p available classes ω_i $(i = 1, 2, \dots, p)$. Let $\mathbf{x}_i$ be a vector containing the lexicographically ordered values of the i-th block to be classified. A neural network classifier is set up to produce a p-dimensional output vector

$$\mathbf{y}(\mathbf{x}_i) = [P^i_{\omega_1} \dots P^i_{\omega_p}]^T \tag{45}$$

where $P^i_{\omega_j}$ denotes the probability that the i-th image block belongs to the j-th class.

Let us first consider that a neural network has been initially trained to perform the previously described classification task using a specific training set $S_b = \{(\mathbf{x'}_1, \mathbf{d'}_1) \dots (\mathbf{x'}_{m_b}, \mathbf{d'}_{m_b})\}$, where vectors $\mathbf{x'}_i$ and $\mathbf{d'}_i$ with $i = 1 \dots m_b$ denote, respectively, the i-th input training vector (image block), and the corresponding desired output vector consisting of p elements. Let $\mathbf{y}(\mathbf{x}_i)$ denote the network output when applied to the i-th block of an image outside the training set. Whenever a change of the environment occurs, new network weights should be estimated through a retraining procedure, taking into account both the former network knowledge and the current situation.

Let us consider retraining in more detail. Let $\mathbf{w}_b$ include all weights of the network before retraining, and $\mathbf{w}_a$ the new weight vector which is obtained through retraining. A training set S_c is assumed to be extracted from the current operational situation composed of m_c image blocks; $S_c = \{(\mathbf{x}_1, \mathbf{d}_1) \dots (\mathbf{x}_{m_c}, \mathbf{d}_{m_c})\}$ where $\mathbf{x}_i$ and $\mathbf{d}_i$ $(i = 1 \dots m_c)$ correspond to the i-th input and desired output retraining data. The retraining algorithm is activated whenever a change of the environment is detected (Section 7.4) and computes the new network weights $\mathbf{w}_a$ by minimizing

$$E_a = E_{c,a} + \eta E_{f,a} \tag{46}$$

$$E_{c,a} = \frac{1}{2} \sum_{i=1}^{m_c} \|\mathbf{z}_a(\mathbf{x}_i) - \mathbf{d}_i\|_2 \tag{47}$$

$$E_{f,a} = \frac{1}{2} \sum_{i=1}^{m_b} \|\mathbf{z}_a(\mathbf{x'}_i) - \mathbf{d'}_i\|_2 \tag{48}$$

where $E_{c,a}$ is the error corresponding to the training set S_c ('current' knowledge), $E_{f,a}$ the corresponding error over the training set S_b ('former' knowledge); $\mathbf{z}_a(\mathbf{x}_i)$ and $\mathbf{z}_a(\mathbf{x'}_i)$ are the outputs of the retrained network with weights $\mathbf{w}_a$. Similarly $\mathbf{z}_b(\mathbf{x}_i)$ would represent the output of the network, consisting of weights $\mathbf{w}_b$. During retraining, initially $\mathbf{z}_b(\mathbf{x}_i)$ is identical to $\mathbf{y}(\mathbf{x}_i)$. Parameter η is a weighting factor accounting for the significance of the current training set compared to the former one and $\|.\|_2$ denotes the L_2-norm.

In real life image or video analysis applications, training set S_c can be a priori unknown; in this case an estimate of S_c should be provided to the system. Apart from the retraining algorithm, a "change detection" decision mechanism is included in it.

Decision Mechanism The goal of this module is to determine when retraining should be activated, or equivalently, to detect the time instances when a change of the environment occurs. Whenever network performance is considered satisfactory (no change of the environment occurs), the network weights and structure remain the same. Instead, new network weights are calculated when network performance is not appropriate, by activating the retraining algorithm. Investigation of the performance of the initially trained network on the current data is the basis of the mechanism, as described in section 7.4.

7.3 The retraining technique

7.3.1 The network architecture

Let us, for simplicity, consider a two-class classification problem, where classes ω_1, ω_2 refer to, for example, foreground and background objects in an image. A feedforward neural network classifier with a single output neuron and a sigmoidal activation function, one hidden layer with q neurons and hyperbolic tangent activation functions, and image blocks of J pixels at its input is used. Extension to classification problems and networks of higher complexity can be performed in a similar way. The input layer weight matrix is denoted $\mathbf{W}^0$ and the output layer weight vector is $\mathbf{w}^1$.

7.3.2 The retraining algorithm

The goal of the training procedure is to minimize the error criterion of Eq. 46 and estimate the new network weights $\mathbf{w}_a$. Let us first assume that a small perturbation of the network weights $\mathbf{w}_b$ (before retraining) is enough to achieve good classification performance. Then,

$$\mathbf{W}_a^0 = \mathbf{W}_b^0 + \Delta\mathbf{W}^0 \tag{49}$$

$$\mathbf{w}_a^1 = \mathbf{w}_b^1 + \Delta\mathbf{w}^1 \tag{50}$$

where $\Delta\mathbf{W}^0$ and $\Delta\mathbf{w}^1$ represent small fluctuations. This assumption leads to an analytical and tractable solution for estimating $\mathbf{w}_a$, since it permits linearization of the non-linear activation function of the neuron, using a first order Taylor series expansion.

Equation 46 indicates that the new network weights are estimated taking into account both the current and the previous network knowledge. To stress, however, the importance of current training data, one can replace Eq. 47 by the constraint that the actual network outputs are equal to the desired ones, that is

$$z_a(\mathbf{x}_i) = d_i \quad , \; i = 1 \dots m_c \tag{51}$$

This means that the first term of Eq. 46, corresponding to error $E_{c,a}$, takes values close to zero, after estimating the new network weights. Through linearization, solving (51) with respect to the weight increments is equivalent to a set of linear equations

$$\mathbf{c} = \mathbf{A}\Delta\mathbf{w} \tag{52}$$

with $\Delta\mathbf{w}$ a vector corresponding to all weight differences, $\mathbf{A}$ a matrix appropriately expressed in terms of the previous network weights, and

$$\mathbf{c} = [z_a(\mathbf{x}_1) \dots z_a(\mathbf{x}_{m_c})]^T - [z_b(\mathbf{x}_1) \dots z_b(\mathbf{x}_{m_c})]^T \tag{53}$$

$$= [d_1 \dots d_{m_c}]^T - [z_b(\mathbf{x}_1) \dots z_b(\mathbf{x}_{m_c})]^T \tag{54}$$

Equation 52) is valid only when the weight changes $\Delta\mathbf{w}$ are small. It can be shown that, given a tolerated error value, proper bounds θ and ϕ can be computed for the weight changes, for each input vector $\mathbf{x}_i$ in S_c:

$$|\mathbf{x}_i^T \Delta\mathbf{w}_k^0| \leq \theta(\mathbf{x}_i), \quad k = 1 \dots q \tag{55}$$

$$|\mathbf{a}(\mathbf{x}_i)^T \Delta\mathbf{w}| \leq \phi(\mathbf{x}_i) \tag{56}$$

where $\mathbf{a}$ is a vector depending on the former weights and on the input vector. Equations 55 and 56 should be satisfied for all $\mathbf{x}_i \in S_c$, so as to assure that each linear equation in Eq. 52 is valid within the given tolerated error. A less strict approach would require that the mean value of the inner products in Eqs. 55-56, for all $\mathbf{x}_i$, be smaller than the mean value of the bounds.

The size of vector $\mathbf{c}$ is smaller than the number of unknown weights $\Delta\mathbf{w}$, since in general a small number of training data is chosen, either through a clustering or a principal component analysis technique. Thus, many solutions exist for Eq. 52, since the number of unknowns is much greater than the respective number of equations. Uniqueness, however, is imposed by an additional requirement, which takes into consideration the previous network knowledge. Among all possible solutions that satisfy Eq. 52 the one which causes a minimal degradation of the previous network knowledge is selected as the most appropriate. It is assumed that the network weights before retraining $\mathbf{w}_b$ have been estimated as an optimal solution over data of set S_b. Furthermore, the weights after retraining provide a minimal error over all data of the current set S_c. Thus, minimization of the second term of Eq. 46, which expresses the effect of the new network weights over data set S_b, can be considered as minimization of the absolute difference of the error over data in S_b with respect to the previous and the current network weights. This means that the weight increments are minimally modified, resulting in the error criterion

$$E_S = \|E_{f,a} - E_{f,b}\|_2 \tag{57}$$

with $E_{f,b}$ defined similarly to $E_{f,a}$ in Eq. 48. It can be shown that Eq. 57 takes the form of

$$E_s = \frac{1}{2}(\Delta\mathbf{w})^T\mathbf{K}^T\mathbf{K}\Delta\mathbf{w} \tag{58}$$

where the elements of matrix $\mathbf{K}$ are expressed in terms of the previous network weights $\mathbf{w}_b$ and the training data in S_b. Thus, the problem results in minimization of Eq. 58 subject to constraints (52) and (55-56).

The number of data belonging to set S_c is in general much smaller than the number of network weights. This is due to the fact that these data have been selected from the current condition using a clustering or principal component analysis technique to discard data of similar content and obtain more reliable solutions. Thus, network overfitting could arise if only these current data were taken into account. However, in our method, the network weights are estimated using data from both the current set S_c and the previous set S_b. The number of elements of S_b is in general greater than the number of network weights since this set includes all a priori network knowledge. Consequently, overfitting issues are not encountered in the proposed retraining algorithm, since the number of training data in S_c and S_b, is greater than the number of network weights.

The error function defined by Eq. 58 is convex, the constraints in Eq. 52 are linear equalities, while the constraints in Eqs. 55-56 are linear inequalities. Thus, the solution should lie on the hyper-surface defined by (52), satisfy the inequalities (55-56) and simultaneously minimize the error function (58). A variety of methods can be used to estimate the weight increments for this minimization problem. In this paper we adopt the gradient projection method, which is presented in Section 7.3.3.

The decision mechanism ascertains whether a small adaptation of the network weights is sufficient to provide accurate classification. In case that such a small adaptation is not adequate, the above training algorithm cannot be used, since the activation functions cannot be approximated by a first order Taylor series. In this case, neither Eq. 51 nor Eq. 57 can be expressed in a simple form and thus the new network weights should be estimated through conventional minimization of Eq. 46 by applying, for example, the backpropagation algorithm. Since both the current and previous network training data sets are taken into account for updating the weights, overfitting issues are not encountered in this case too.

Each time the decision mechanism ascertains that retraining is required, a new training set S_c is created, which represents the current condition. Then, new network weights are estimated taking into account both the current information (data in S_c) and the former knowledge (data in S_b). Since the set S_c has been optimized over the current condition, it cannot be considered suitable for future states of the environment. This is due to the fact that data obtained from future states of the environment may be in conflict with data obtained from the current one. In contrast, it is assumed that the training set S_b, which is in general provided by a vendor, is able to roughly approximate the desired network performance in any state of the environment. Consequently, in every network retraining phase, a new training set S_c is created and the previous one is discarded, while new weights are estimated based on the current set S_c and the old one S_b, which remains constant throughout network operation.

7.3.3 The gradient projection method

The philosophy of the gradient projection method is, starting from a feasible point, to move in a direction which decreases E_S and simultaneously satisfies the constraints (52) and (55-56). Let us give some definitions first. A point is called feasible when it satisfies all the constraints. An inequality is characterized as an active constraint when it is about to be violated at a feasible point, in the sense that it turns to equality at this point. Otherwise the inequality is an inactive constraint. For example, the inequality $\Delta\mathbf{w}^T\mathbf{a}(\mathbf{x}_i) \leq \phi(\mathbf{x}_i)$ in (56) becomes active at a feasible point $\Delta\mathbf{w}$ when $\Delta\mathbf{w}^T\mathbf{a}(\mathbf{x}_i) = \phi(\mathbf{x}_i)$ at this current point. Otherwise it is considered to be inactive. In the following q_a denotes the number of active inequalities. By convention we consider any equality constraint to be active at a feasible point. Thus, the total number of active constraints, i.e., equalities and active inequalities, is equal to $m_c + q_a$.

Only the active constraints define the direction which causes a decrease of function E_S. As a result, a working set is formed at each feasible point which includes all active constraints. Then, the method tries to move in a direction which decreases the error function E_S while keeping the working constraints active. This is accomplished by projecting the negative gradient of E_S onto the subspace that is tangent to the surface defined by these constraints. If the resulting vector is non zero, it determines the direction for the next step. To compute this projection, let matrix $\mathbf{A}_q$ be composed of the above-defined working set. Matrix $\mathbf{A}_q$ contains all rows of matrix $\mathbf{A}$ (since equalities are always active at any feasible point) and additional rows which include the active inequalities. Thus, $\mathbf{A}_q$ is of $(m_c + q_a) \times N_w$, where we recall that N_w corresponds to the total number of network weights. Let us denote by $\mathbf{h}(n)$ the projection of the negative gradient of E_S in the n-th iteration step of the algorithm. Then, adaptation of the weight increments is made as follows:

$$\Delta\mathbf{w}(n+1) = \Delta\mathbf{w}(n) + \mu(n)\mathbf{h}(n) \tag{59}$$

where $\mu(n)$ is a scalar that determines the rate of convergence. We can estimate vector $\mathbf{h}(n)$ as

$$\mathbf{h}(n) = -\mathbf{P}\nabla E_S = -\mathbf{Q}\Delta\mathbf{w} \tag{60}$$

with the projection matrix $\mathbf{P}$ and the matrix $\mathbf{Q}$ according to

$$\mathbf{P} \;=\; \mathbf{I} - \mathbf{A}_q^T(\mathbf{A}_q\mathbf{A}_q^T)^{-1}\mathbf{A}_q \tag{61}$$

$$\mathbf{Q} = \mathbf{P}\mathbf{K}^T\mathbf{K}\Delta\mathbf{w} \tag{62}$$

Starting from an initial feasible point, the gradient projection algorithm computes the next feasible point based on (59) until the projected gradient $\mathbf{h}(n)$ is close to zero. Now, it is possible to improve the solution by relaxing one active constraint, i.e., one active inequality. This happens since the Kuhn-Tucker conditions are not satisfied for the original. A vector $\mathbf{g}$ is calculated by

$$\mathbf{g} = -(\mathbf{A}_q\mathbf{A}_q^T)^{-1}\mathbf{A}_q\nabla E_S \tag{63}$$

Each element of $\mathbf{g}$ corresponds to an active constraint. In case that there are negative elements of $\mathbf{g}$, the Kuhn-Tucker conditions are not satisfied and a new improved feasible solution can be obtained by relaxing the respective constraint from the working set. This is performed by deleting the corresponding row of $\mathbf{A}_q$, so that the respective constraint becomes inactive. In this case the projection matrix $\mathbf{P}$ should be recomputed since matrix $\mathbf{A}_q$ has modified. A new iteration then commences until the projected gradient $\mathbf{h}(n)$ reaches a value close to zero. If all elements of $\mathbf{g}$ are nonnegative, the Kuhn-Tucker conditions are satisfied and the process is terminated since it is not possible to further improve the obtained solution. The number of times that the working set is changed is equal to number of inequalities; equalities are always considered active. Thus, the number of iterations required for getting all elements of $\mathbf{g}$ positive so that the Kuhn-Tucker conditions are satisfied, is bounded by the number of inequalities.

7.4 The decision mechanism for network retraining

The purpose of this mechanism is to detect when the output of the neural network classifier is not appropriate and consequently to activate the retraining algorithm at those time instances when a change of the environment occurs. Images or video frames are denoted by $\mathbf{x}(k, N)$, where k is the image number (time), and N denotes the retraining instance. Index k is reset each time retraining takes place, with $\mathbf{x}(0, N)$ corresponding to the image where the N-th retraining of the network was accomplished.

Retraining of the network classifier is accomplished at time instances where its performance deteriorates, i.e., the current network output deviates from the desired one. Let us recall that vector $\mathbf{c}$ expresses the difference between the desired and the actual network outputs based on weights $\mathbf{w}_b$ and applied to the current data set S_c. As a result, if the norm of vector $\mathbf{c}$ increases, network performance deviates from the desired one and retraining should be applied. On the contrary, if vector $\mathbf{c}$ takes small values, then no retraining is required. In the following we denote this vector as $\mathbf{c}(k, N)$ indicating its dependence upon image $\mathbf{x}(k, N)$. However, direct calculation of the norm of $\mathbf{c}(k, N)$ would require application of the MAP estimation procedure described in the previous section so as to extract from each image . In video processing applications, for example, each image frame arrives at a rate of 40msec (25 frames/sec in PAL system); it will therefore be very time consuming to activate the MAP estimation procedure for each $\mathbf{x}(k, N)$. In such applications, detection of the retraining time instances can be performed through the following methodology.

7.4.1 Detection of the retraining time instances

Let us assume that the N-th retraining phase of the network classifier has been completed. If the classifier is then applied to all blocks of the image $\mathbf{x}(0, N)$, including the ones used for retraining, it is expected to provide classification results of good quality. The difference between the output of the retrained network and of the initially trained classifier at image $\mathbf{x}(0, N)$ constitutes an estimate of the level of improvement that can be achieved by the retraining procedure. Let us denote by $e(k, N)$ this difference:

$$e(k, N) = \frac{1}{L}\sum_{i=1}^{L} \|\mathbf{y}(\mathbf{x}_i(k, N)) - \mathbf{z}(\mathbf{x}_i(k, N))\|_2 \tag{64}$$

where L is the number of blocks in the image. It is anticipated that the level of improvement expressed by $e(k, N)$ will be close to that of $e(0, N)$ as long as the classification results are good. This will occur when input images are similar, or belong to the same scene with the ones used during the retraining phase. A change in the error is generally due to a change of the environment. Thus, the quantity $a(k, N) = |e(k, N) - e(0, N)|$ can be used for detecting the change of the environment or equivalently the time instances where retraining should occur. In particular, if $a(k, N)$ is smaller than some threshold T no retraining is needed. In case of retraining, index k is reset to zero while index N is incremented by one.

7.4.2 The retraining phase

When $a(k, N)$ exceeds threshold T at time k_0, the $(N + 1)$-th retraining phase starts. In this case, the training set S_c which represents the current condition should be generated; this can be performed in different ways depending on the application, e.g., manually, or through a MAP estimation procedure mentioned in the beginning of the paper. Vector $\mathbf{c}(k_0, N) \equiv \mathbf{c}(0, N + 1)$ is also calculated. The weights of the network before retraining are used as previous network weights $\mathbf{w}_b$ in the retraining algorithm.

The training algorithm provides the new network weights through minimization of (58) subject to constraints (52) and (55-56). However, existence of a feasible solution is assured only if the hyper plane given by (52) crosses through the boundary constraints imposed by (55-56). For this reason the minimal distance from the origin to the surface $\mathbf{c}(0, N+1) - \mathbf{A}\Delta\mathbf{w} = \mathbf{0}$, is first calculated, by solving the following minimization problem with respect to $\Delta\mathbf{w}$:

$$\text{Minimize } \|\Delta\mathbf{w}\|_2 \text{ subject to } \mathbf{c}(0, N + 1) - \mathbf{A}\Delta\mathbf{w} = \mathbf{0} \tag{65}$$

Using Lagrange multipliers the above minimization problem leads to the following solution

$$\Delta\mathbf{w}^p = \mathbf{A}(\mathbf{A}^\mathbf{T}\mathbf{A})^{-1}\mathbf{c}(0, N + 1) \tag{66}$$

where $\Delta\mathbf{w}^p$ is the minimal distance from the origin to $\mathbf{c}(0, N + 1) - \mathbf{A}\Delta\mathbf{w} = \mathbf{0}$. For this value of $\Delta\mathbf{w}^p$, if the inequality constraints defined by (55-56) are satisfied for all data in S_c, the new network weights can be calculated by minimizing (58) subject to (52) and (55-56). Otherwise, Eq. 46 cannot be expressed in the simple form of (52) and (58) and thus direct minimization of (46) is required for estimating the network weights. This is due to the fact that the former knowledge is quite different from the current one and a small weight perturbation is inadequate for sufficiently adapting the network weights. Another situation where the above can be used is the following. Let us assume that several neural network classifiers along with their respective training sets and weights have been given to a user by different suppliers or that the user has created different retraining sets himself using the procedure presented in the paper; the user is assumed to apply one of them. In cases that the decision mechanism detects that retraining is needed, but the training algorithm cannot be applied to the users' classification problem since the boundary constraints are not satisfied, it seeks, among the available networks, the most appropriate one for the current environment. Then, the selected network is considered to represent the best previous knowledge and the training algorithm is used to further improve the network performance.

8 Application: emotion classification based on vocal cues

The techniques of the PHYSTA project are applied to the automatic recognition of emotions based on facial and vocal cues. At the time of the production of this report the PHYSTA emotion database had not been completed yet. Therefore, speech data recorded by Sinead McGilloway as part of her PhD research at Queens University of Belfast (Belfast, United Kingdom) was used for the results reported in this report. In the study of McGilloway 40 subjects read a neutral passage and simulated four emotions, namely fear, happiness, sadness and anger. After recording the speech fragments were processed by the ASSESS system. The ASSESS speech processing system was developed at Queens University of Belfast and extracts a large number of statistical figures regarding amplitude variations, fundamental frequency characteristics and spectral information.

Since the total number of speech features provided by ASSESS (about 400) is far too large for the classification task, two relevant feature sets were a priori chosen from the total data set. Set S_1 contains 32 features and were chosen by McGilloway based on a statistical analysis. Set S_2 contains 20 features which were chosen on the basis of expert knowledge. A total number of 196 feature vectors and corresponding emotions were available for training and testing of the neural network classification systems. This total data set was used for the generation of 10 random divisions in 70 % training and 30 % test data. The test data were in no way used in the training process. Since the size of the training sets (137 data pairs) is about equal to the number of network parameters in the classification systems, it is clear that generalisability of the classifiers for independent data may be problematic.

8.1 Bayesian classification

8.1.1 Methods

For the emotion classification task a MLP with 'softmax' outputs y_i and a cross-entropy error function G are appropriate:

$$y_i = \frac{e^{a_i}}{\sum_j e^{a_j}} \tag{67}$$

$$G = \sum_{m=1}^{L} \sum_{i=1}^{5} t_i^m \log y_i(\mathbf{x}^m; \mathbf{w}) \tag{68}$$

with $t_i^m \in \{0, 1\}$ a target label specifying the emotion class, $\mathbf{x}^m$ an input vector, $\mathbf{w}$ the network parameters (weights and thresholds), and the activation $\mathbf{a}$ being the resultant of the processing of the first two weight layers:

$$\mathbf{a} = \mathbf{W_2 f}(\mathbf{W_1 x} + \mathbf{w_0}) \tag{69}$$

with $\mathbf{f}(\mathbf{x})$ an array of sigmoidal functions. Five hidden units were applied in the classification systems. In the Bayesian context, the posterior distribution of the network parameters given the data is determined by the cross-entropy function G and a regularization term E_W:

$$p(\mathbf{w}|D) = \frac{1}{Z_S}\exp(-G - \alpha E_W) \tag{70}$$

$$E_W = \frac{1}{2}\sum_i w_i^2 \tag{71}$$

with Z_S a normalization constant and α a regularization constant. As an extension to this scheme, different regularization constants α_j may be used for different groups of network parameters. This approach is applied in automatic relevance determination (ARD) where a unique regularization parameter α_j is associated with the weigths in the input layer for each input signal.

During training both the network parameters $\mathbf{w}$ and the regularization parameter(s) α are optimized iteratively. First $\mathbf{w}$ is optimized using some gradient descent technique. Then, the regularization parameter (singular for simplicity) is adapted assuming a Gaussian posterior probability distribution:

$$p(\mathbf{w}|D,\alpha) \approx \frac{1}{Z_M}\exp\{-G(\mathbf{w}^{mp}) - E_W(\mathbf{w}^{mp}) - \frac{1}{2}(\mathbf{w}-\mathbf{w}^{mp})^T\mathbf{H}(\mathbf{w}-\mathbf{w}^{mp})\} \tag{72}$$

with $\mathbf{w}^{mp}$ the (most probable) weights following from the gradient optimization and $\mathbf{H}$ the corresponding Hessian matrix. It follows from (MacKay, 1996), that maximum evidence is attained by choosing the regularization parameter according to the iterative scheme

$$\gamma = W - \alpha \mathrm{Trace}(\mathbf{H}^{-1}) \tag{73}$$

$$\alpha^{mp} = \frac{\gamma}{\sum_i (w_i^{mp})^2} \tag{74}$$

with W the total number of network parameters.

8.1.2 Classification for feature sets S_1 and S_2

For the two feature sets S_1 and S_2, classification rates on the 10 test sets of, respectively, $r_1 = 53.1 \pm 7.3\%$ and $r_2 = 37.6 \pm 4.1\%$ were attained. The standard deviation of the classification rate is primarily due to the random divisions of the data in the training and test set and to a small extent due to the initial weight configurations. In the 10 data sets for features S_1 classification rates ranging from 40.0 to 64.4% were attained. The classification rate for feature set S_2 is clearly inferior to the rate of set S_1, indicating that this feature set does not contain sufficient information relevant for the classification. It will therefore not be considered anymore.

Figure 1 shows the confusion matrix of the classification based on set S_1. Although all emotions are classified correctly above chance level, especially sadness and anger are well distinguished by the system. McGilloway did not evaluate the perception of the speech traces and so the classification rate of the neural network can not be compared directly with human emotional discrimination capability. Other studies of Van Bezooijen (1984) and Scherer et al. (1991) show recognition rates mostly in the range from 50 to 75% for a limited set of emotions (Table 1). In accordance with the empirical data of both studies sadness and anger are best recognized. Fear is recognized by the NN in 46% of the cases which is somewhat less than the empirical rates. Happiness is distinguished clearly poorer by the NN than by human observers. A neutral emotional state was not used in the empirical studies. It stands out that the confusion with the other emotions is about equal for all other four emotions (although deviations are high), indicating that the neutral state indeed is an average state. The average recognition rates of the four emotions (fear, happiness/joy, sadness, anger) which were used both in the empirical studies and this NN study are, respectively, 65% and 55%. The large standard deviations of the elements of the confusion matrix indicate that the classification performance of the networks strongly depends on the arbitrary divisions in test and training data. This indicates that the data are rather heterogeneous and that the number is training data is relatively small.

	afraid	happy	neutral	sad	angry
afraid	45.8 +/− 11.9	8.3 +/− 11.8	15.0 +/− 11.0	14.2 +/− 9.7	16.7 +/− 14.7
happy	5.0 +/− 4.3	40.0 +/− 12.3	16.7 +/− 7.9	21.7 +/− 10.5	16.7 +/− 9.6
neutral	13.3 +/− 5.8	16.7 +/− 13.6	45.0 +/− 15.3	14.2 +/− 11.1	10.8 +/− 11.1
sad	9.1 +/− 8.6	6.4 +/− 4.4	11.8 +/− 9.6	72.7 +/− 11.3	0.0 +/− 0.0
angry	15.0 +/− 8.6	7.5 +/− 8.3	13.3 +/− 9.0	0.8 +/− 2.6	63.3 +/− 16.3

Figure 1: *Confusion matrix showing the mean classification plus standard deviation of the 5 emotional states by the Bayesian network (set S_1). Rows indicate the actual emotions, columns the classification.*

Table 1: *Classification rates of human observers for two studies.*

Study	Fear	Disgust	Joy	Sadness	Anger
Bezooijen (1984)	58	49	72	67	74
Scherer et al. (1991)	52	28	59	72	68

8.1.3 Application of confidence levels

For the classification in the former section the emotion with the highest probability score was chosen without considering the probability rates of the other emotions. The classification may be refined by requiring a minimum difference in the probability between the two highest probability estimates. This means the classification process may now render a 'do not know' state if the difference between the probabilities of two emotions is too low. The results of this procedure for the classifier based on set S_1 are shown in Fig. 2. A strong rise in the number of cases which can not be classified can be observed. For example, about 20% of the cases are not classified for a probability difference of 0.1. While exclusion of the low confidence cases leads to a clear increase in the performance on the training set from 81.6% to 89.5% with 18.3% 'do not knows' for $\Delta p = 0.1$, the average performance increase on the test set is much less impressive: from 53.1% to 56.0% with 22.9% 'do not knows'. Only if the number of cases to be classified becomes very small, classification rates above 80% are achieved for the test data. The slow increase of the classification rate as function of the a posteriori minimum probability difference for the independent test data indicates that there is a large variety in the vocal cues which could not be captured by the network, because of the relatively small size of the training set.

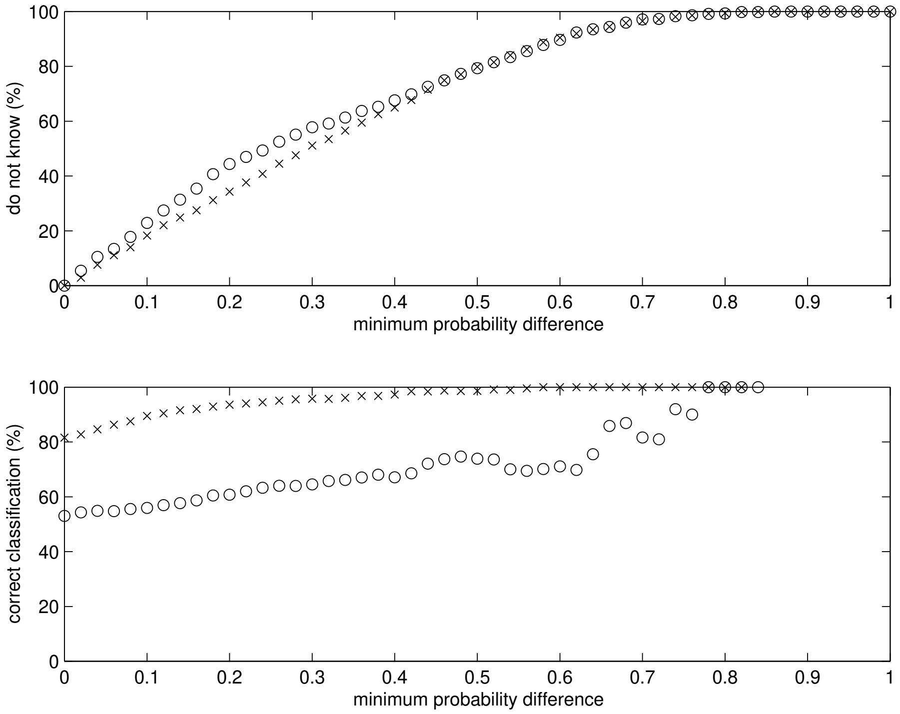

Figure 2: *Results of classification requiring a minimum estimated probability difference for both test data (circles) and training data (crosses). Top: percentage of cases which can not be classified. Bottom: percentage of correctly classified emotions.*

8.1.4 Automatic relevance determination of speech features

In the Bayesian framework automatic relevance determination (ARD) can be used to assess the relevance of the input signals. This is implemented by addressing an individual regularization parameter to each input signal. Application of the ARD procedure to set S_1 resulted in a classification rate of $53.4 \pm 6.3\%$, which is similar to the rate attained with one global regularization parameter. Although the classification rates converged during the re-estimation process, the regularization parameters tended to fluctuate from one re-estimation cycle to the other. Nevertheless, some regularization parameters were clearly higher. Average measures of the regularization parameters were attained by normalizing the regularization vector in each cycle and averaging this normalized vector over the last 7 re-estimation cycles, where no change in the classification rate was manifested anymore. These average values are depicted in Fig. 3.

It follows from this figure that there exists a clear division between two groups of, respectively, irrelevant and relevant features with relatively high and low regularization parameters. Excluding the features with $\alpha > 2\ 10^{-3}$ (dashed line in Fig. 3) there remain 18 relevant features, which are described in Table 2.

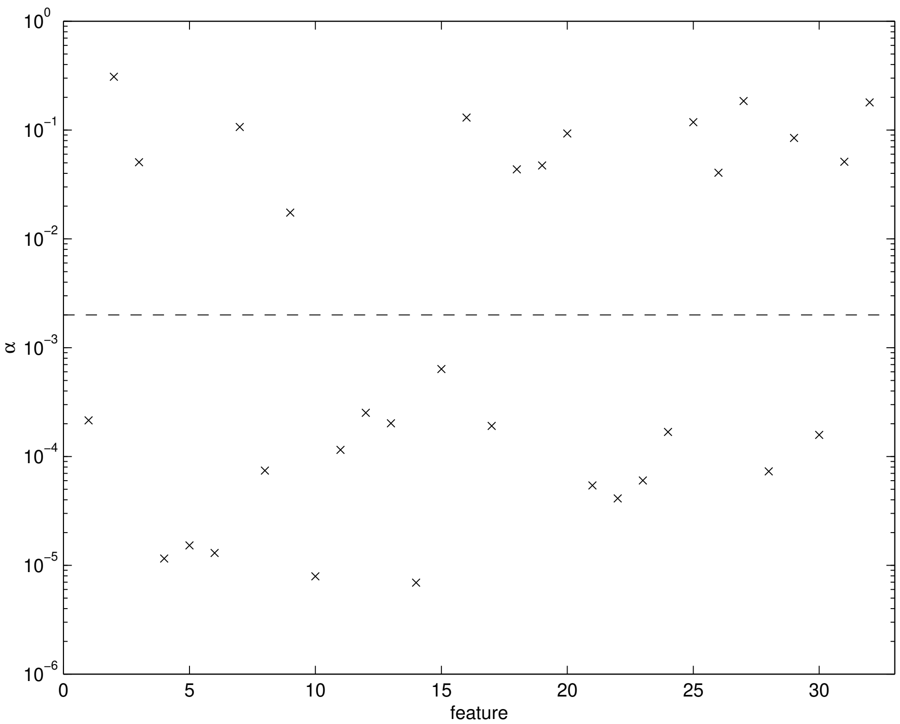

Figure 3: *Normalized and averaged regularization parameters α for each feature in set S_1 attained after an ARD procedure. Features with α above the dashed line are deemed irrelevant.*

To test whether no relevant features were removed unjustly the system was trained with the subset of 18 features of S_1 and one global regularization parameter, like in Section 8.1.2. Using the subset a classification rate equal to the full set rate could be attained: $52.5 \pm 5.9\%$. The corresponding confusion matrix is depicted in Fig. 4. It is similar to the one of the full data set in Fig. 1. The average correct classification of happiness is now smaller, but not significantly.

8.2 Classification by network ensembles

8.2.1 Methods

Ensembles of MLP classifiers with soft-max output functions trained on bootstrap sets of the training sets were used to attain a good overall classification performance and to estimate confidence intervals (see Section 5 for some background). The structure of each MLP was equal to the network used in the Bayesian context (Section 8.1.1). The results of all networks in the ensemble were weighted equally, in other words, the overall output is the mean of all outputs. The training procedure applied bootstrap samples from the training sets. No regularization parameter was included, but an early stopping procedure was used. The classification performance on the training set members which were not in the bootstrap sample was taken as a measure for the performance on independent data. Since it was observed that the classification

Table 2: *ASSESS features which are deemed relevant for emotion classification by the ARD procedure. See also the ASSESS manual for the exact meaning of the variables. The feature numbers refer to Fig.3.*

number	description
1	Duration of a tune
4	Average energy in F0 region
5	Mean of all amplitude points
6	Median of all amplitude points
8	Mean of amplitude maxima
10	Mean of amplitude minima
11	Inter-quartile (IQR) of amplitude minima
12	IQR of amplitude rise heights
13	IQR of amplitude fall heights
14	Number of fundamental frequency (F0) points
15	Mean of all F0 points
17	IQR F0 maxima
21	IQR F0 rise heights
22	Median amplitude rise duration
23	Median amplitude fall duration
24	IQR Amplitude of top plateaux
28	Median duration of F0 silences
30	Median duration of F0 falls

rate could have multiple maximums during learning, the network with the best performance attained in 50 conjugate gradient iterations was taken to be optimum.

8.2.2 Results

The attained classification performance of an ensemble of 30 networks for feature set S_1 was $48.6 \pm 5.9\%$, somewhat lower than the rate attained in the Bayesian framework. Figure 5 shows the classification rate as function of the number of networks in the ensemble. Clearly, the performance improves with the ensemble size, as was expected from the theoretical background discussed in Section 5. Furthermore, it stands out that the performance on the training data shows a stronger improvement than the classification on the independent test data. The confusion matrix for the ensemble predictions is depicted in Fig. 6.

Instead of the maximum a posteriori probability classification, the minimum a posteriori difference classification may be applied. Its results are shown in Fig. 7. A fast increase in the number of 'do not knows' in association with a slow increase in the classification rate for the test data is manifested. The results are similar to those of its Bayesian competitor depicted in Fig. 2.

A potential advantage of the bootstrapping procedure is the estimation of the standard errors on its predictions. Assuming a Gaussian distributions, the means and standard deviations enable the classification with a minimum estimated significance level. The results of this procedure are depicted in Fig. 8. A slow increase of the classification rate as function of the significance level can be observed. Even for a significance level of $p = 0.98$, meaning that the probability is 0.98 that the highest network output is indeed larger than the second largest

	afraid	happy	neutral	sad	angry
afraid	49.2 +/– 13.9	4.2 +/– 5.9	19.2 +/– 17.1	14.2 +/– 13.1	13.3 +/– 11.2
happy	8.3 +/– 3.9	30.8 +/– 18.9	21.7 +/– 16.8	21.7 +/– 10.5	17.5 +/– 12.7
neutral	11.7 +/– 7.0	20.8 +/– 13.2	44.2 +/– 13.1	12.5 +/– 11.3	10.8 +/– 11.8
sad	10.0 +/– 10.0	4.5 +/– 4.8	9.1 +/– 7.4	75.5 +/– 8.6	0.9 +/– 2.9
angry	14.2 +/– 13.6	6.7 +/– 6.6	14.2 +/– 9.7	0.0 +/– 0.0	65.0 +/– 17.5

Figure 4: *Confusion matrix showing the mean classification plus standard deviation of the 5 emotional states by the Bayesian network based on a subset of S_1 containing 18 features). Rows indicate the actual emotions, columns the classification.*

output, the classification rate is only 69% while in 80% of the cases no confident classification is feasible.

8.3 Discussion of the emotion classification

The classification of five basic emotions based on vocal cues was successful in about half of the cases, being clearly above chance level (20%) but also far from perfect. Unfortunately, no human classification rates were available for the data used in this study, but related studies of Bezooijen (1984) and Scherer et al. (1991) show that human recognition performance is also far from perfect. For the four emotions equal in the empirical studies and the Bayesian NN classification rates of 65% and 55% were attained, respectively. The large variation between the classification rates of systems trained and tested with different divisions of the total data set indicates that (1) there exist distinct differences between vocal cues of the subjects and (2) the size of the training set is too small to represent this broad repertoire. Given the small size of the training sets (137 elements) and the large number of network parameters (ranging from 120 to 190) due to the large number of features (18 to 32), it would indeed have been a surprise if generalisability for independent data would not have been problematic.

This significant variety is also indicated by the slow increase in the classification rate as function of the a posteriori minimum probability difference for the test data.

The Bayesian approach was found to outperform the ensemble method by about 4%. Furthermore, the computation time of Bayesian training compares favorably with the ensemble training, since each network in the ensemble has to be optimized separately. Another advan-

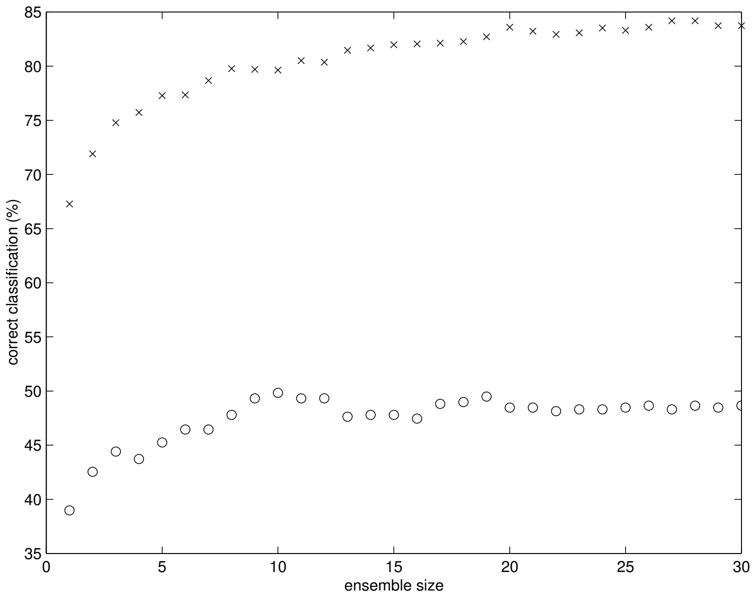

Figure 5: *Classification rate as function of the ensemble size. Crosses represent performance on the training set, circles indicate the test performance.*

tage of the Bayesian approach is support of automatic relevance determination of the input signals. Using this method 14 of the 32 input signals could be removed without a decrease in the recognition rate. In the ensemble approach 'automatic' relevance determination would come down to tedious retraining with a large number of limited sets of input signals. The Bayesian approach thus turns out as the best of the two methods.

References

Bezooijen R van (1984). The characteristics and recognizability of vocal expressions of emotions. Foris, Dordrecht, The Netherlands

Bishop CM (1994). Novelty detection and neural network validation. IEE Proceedings on Vision and Image Signal Processing 141(4):217-222

Bishop CM (1995). Neural networks for pattern recognition. Oxford University Press, Oxford, England

Breiman L (1994). Bagging predictors. Technical report no. 421, Department of Statistics, University of California, Berkeley, USA

Efron B, Tibshirani RJ (1993). An introduction to the bootstrap. Chapman & Hall, New York, USA

Heskes T (1997a). Practical confidence and prediction intervals. In: Advances in Neural Information Processing 9, eds. Mozer M, Jordan M, Petsche T, pp. 176-182

Heskes T (1997b). Balancing between bagging and bumping. In: Advances in Neural Information Processing 9, eds. Mozer M, Jordan M, Petsche T, pp. 466-472

	afraid	happy	neutral	sad	angry
afraid	45.8 +/− 9.0	5.8 +/− 8.8	15.8 +/− 13.3	15.0 +/− 12.3	17.5 +/− 14.4
happy	7.5 +/− 8.3	32.5 +/− 9.2	18.3 +/− 11.7	23.3 +/− 9.5	18.3 +/− 11.0
neutral	7.5 +/− 7.3	15.0 +/− 12.3	46.7 +/− 17.2	16.7 +/− 9.6	14.2 +/− 8.8
sad	11.8 +/− 12.2	8.2 +/− 6.7	14.5 +/− 13.0	63.6 +/− 16.0	1.8 +/− 3.8
angry	18.3 +/− 13.5	6.7 +/− 6.6	14.2 +/− 9.7	5.0 +/− 5.8	55.8 +/− 19.7

Figure 6: *Confusion matrix for ensemble classification.*

Heskes TM, Kappen B (1991). Learning processes in neural networks. Physical Review A 44(4):2718-2726

Krogh A, Vedelsby J (1996). Neural network ensembles, cross validation, and active learning. In In: Advances in Neural Information Processing 7, eds. Tesauro G, Touretzky D, Leen T, pp. 231-238

MacKay DJC (1996). Bayesian methods for backpropagation networks. In: Models of neural networks III, eds. Domany E, Hemmen JL van, Schulten K, Springer, New York, USA

Scherer KR, Banse R, Wallbott HG, Goldbeck T (1991).Vocal cues in emotion encoding and decoding. Motivation and Emotion 15:123-148

Tibshirani R (1996). A comparison of some error estimates for neural network models. Neural Computation 8:152-163

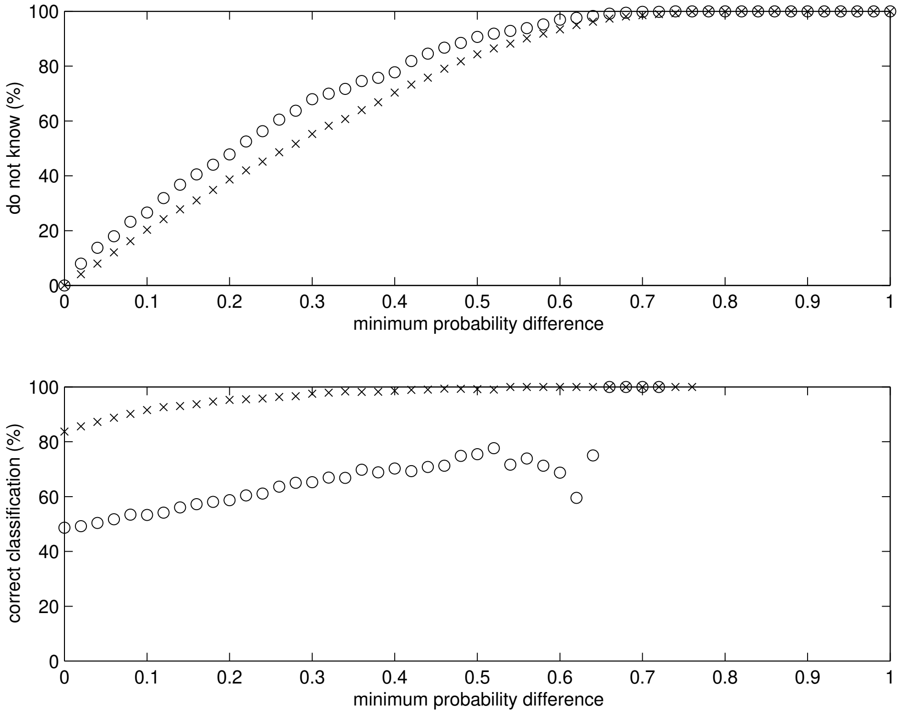

Figure 7: *Results of classification requiring a minimum estimated probability difference for both test data (circles) and training data (crosses). Top: percentage of cases which can not be classified. Bottom: percentage of correctly classified emotions.*

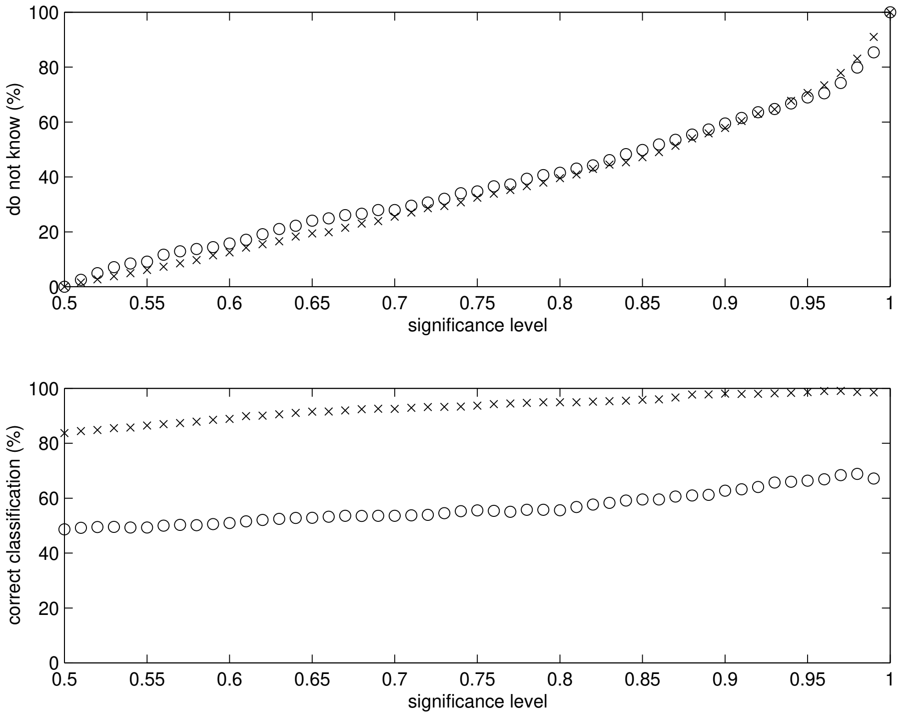

Figure 8: *Results of classification requiring a minimum significance level for both test data (circles) and training data (crosses). Top: percentage of cases which can not be classified. Bottom: percentage of correctly classified emotions.*