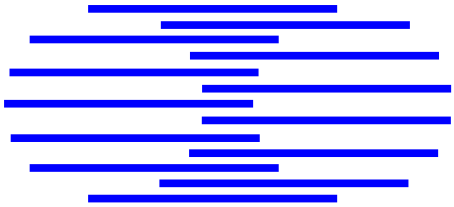
IDIAP RESEARCH REPORT

Dalle Molle Institute
for Perceptual Artificial
Intelligence • P.O.Box 592 •
Martigny • Valais • Switzerland

phone +41 - 27 - 721 77 11
fax +41 - 27 - 721 77 12
email secretariat@idiap.ch
internet http://www.idiap.ch

IDIAP

Martigny - Valais - Suisse



A NEURAL NETWORK FOR CLASSIFICATION WITH INCOMPLETE DATA

Andrew C. Morris

IDIAP-RR 00-23

August 2000

REDUCED VERSION TO APPEAR IN
Int. Conf. of Spoken Language Processing, ICSLP 2000, Beijing

IDIAP-RR 00-23

A NEURAL NETWORK FOR CLASSIFICATION WITH INCOMPLETE DATA

Andrew C. Morris

August 2000

REDUCED VERSION TO APPEAR IN
Int. Conf. of Spoken Language Processing, ICSLP 2000, Beijing

Abstract

If the data vector for input to an automatic classifier is incomplete, the optimal estimate for each class probability must be calculated as the expected value of the classifier output. We identify a form of Radial Basis Function (RBF) classifier whose expected outputs can easily be evaluated in terms of the original function parameters. Two ways are described in which this classifier can be applied to robust automatic speech recognition, depending on whether or not the position of missing data is known.

Acknowledgements: This work was supported by the EC/OFES (European Community / Swiss Federal Office for Education and Science) RESPITE project (REcognition of Speech by Partial Information TEchniques). Recognition tests for the methods presented in this report were carried out in collaboration with the speech group at Sheffield University, U.K.

Contents

1. Introduction 7

2. IDCN architecture 8

2.1 Position of missing data given 8

2.2 Position of missing data unknown 9

3. IDCN training 9

3.1 Parameter initialisation 9

3.2 Error gradient calculation 10

3.3 Gradient descent iteration 11

4. Recognition with missing data 11

5. Summary and conclusion 12

Acknowledgements 12

Appendix A: Using HTK for both Gaussians and output layer weights initialisation 13

Appendix B: Derivation of IDCN error gradient equations 14

Appendix C: Derivation of expected class posterior probabilities 15

References 16

1. Introduction

In any realistic automatic recognition task it is common that part of the input feature vector x to be classified is corrupted by some kind of noise process, and the recognition performance of a system which is not trained to expect this kind of noise will degrade dramatically as the noise level increases. In many cases this problem can be reduced by applying some kind of noise removal or data enhancement process. But there are also many situations in which some feature components are irretrievable. The approach taken in this case depends on to what extent it is possible to identify which features have been corrupted.

If the position of missing features is given, then the estimate for the posterior probability for each class, which is best in the sense that it gives the maximum probability of correct classification, can be obtained as the expected value of the classifier output for that class, conditioned by any available constraints on the missing data [10]. The main problem with this approach is that for most classifiers, the expected value of the class probability outputs cannot be obtained as a simple closed form expression from the classifier parameters.

If the position of missing data is not known, one successful approach [6, 11, 12] has been to train a separate classifier for each possible position of missing data and then to combine the posteriors for one class as a weighted sum over all classifiers. Even with equal weights this approach shows some robustness to missing data, because “uncertain” classifiers tend to contribute equal and therefore small probabilities to each class. The problem with this approach is that the number of different possible positions of missing data is generally far too large to allow training of a separate classifier for each position.

In this paper we present a particular form of Radial Basis Function (RBF) classifier in which the output layer uses Bayes’ Rule to directly transform pooled mixture likelihoods from the RBF layer into a-posteriori class probabilities [2, 3, 8, 17]. Even though the output units are non-linear, the expected outputs of this classifier, for any given missing data components, are a simple function of the original classifier parameters. The use of closely related RBF networks for recognition with missing data is not new [1], but to the author’s knowledge the particular form of incomplete data classification network (IDCN) described here has not been used before in either of the techniques presented in this report.

In Section 2 we present the IDCN architecture, and describe how it can be applied in two different kinds of HMM/ANN hybrid system for automatic speech recognition (ASR), depending on whether the position of missing data is known, or otherwise. In Section 3 we describe various ways in which the IDCN can be trained for ASR. Section 4 shows how network outputs (class posterior probabilities) are calculated when some of the input features are missing. In Section 5 the work is summarised, problems arising are briefly discussed and new ways forward are suggested.

2. IDCN architecture

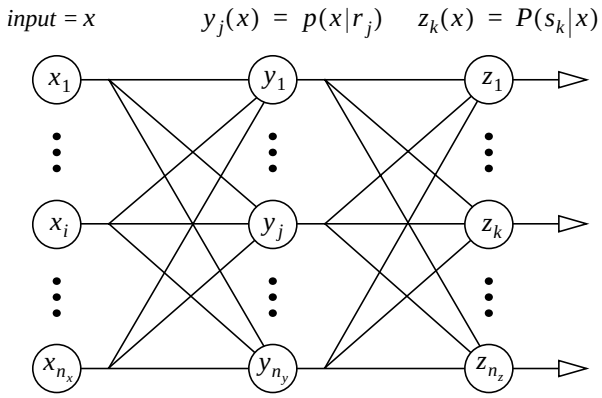


Figure 1: *RBF network used here for classification with incomplete data. The output layer uses Bayes' Rule to directly transform pooled mixture likelihoods from the RBF layer into a-posteriori class probabilities.*

The network has one input, one hidden and one output layer, as shown in Fig.1. Each RBF unit y_j in the hidden layer uses a diagonal covariance Gaussian $y_j(x)$ to model the probability density $p(x|r_j)$ for input vector x having been generated by this Gaussian, while each output unit uses a function $z_k(x)$ to model the posterior probability that x is from output class k. If r_j denotes that x was generated by Gaussian j, and s_k that x is from class k, then:

$$y_j(x) = p(x|r_j) = N(x, \mu_j, v_j) \tag{1}$$

$$z_k(x) = P(s_k|x) = \frac{p(x, s_k)}{p(x)} = \frac{net_k}{p(x)} \tag{2}$$

where

$$net_k = \sum_j P(x, r_j, s_k) = \sum_j P(r_j, s_k)\, p(x|r_j, s_k) = \sum_j P(r_j, s_k)\, p(x|r_j) = \sum_j w_{jk} y_j \tag{3}$$

$$p(x) = \sum_{j,k} p(x, r_j, s_k) = \sum_k net_k \tag{4}$$

Although the above structure of the IDCN does not change, the way in which it is applied depends on whether the position of missing input data is known.

2.1 Position of missing data given

The IDCN can be used as a front end to a conventional HMM based ASR system, whereby the log likelihoods which are normally calculated from the Gaussian mixture models for each hidden state are replaced, during decoding, by log scaled likelihoods from the IDCN (by dividing by their class priors $P(s_k)$, then taking the logarithm). This comprises a form of HMM/ANN based ASR system [3] which is suitable for use with missing data when the position of missing data is given. The main potential advantages of this model over the purely HMM based missing-data theory based system described in [10] is that the ANN is discriminatively trained and provides a more powerful model for capturing spectral dynamics.

2.2 Position of missing data unknown

In principle a single IDCN can be used to replace the 2^d different ANN experts which are normally required [12] to cover all possible selections of missing features from a d dimensional feature vector. Provided that the combined features input to each ANN expert are merely concatenated (i.e. no compression, orthogonalisation, or whatever is applied), the expected posteriors for each position of missing features can be computed directly from the IDCN parameters, and then simply combined in a linearly weighted sum [11] or geometrically weighted product [5].

3. IDCN training

Classifier parameters to be trained are the mean and variance vectors in Eq.(1) for each Gaussian RBF unit, and the output layer weights, w, in Eq.(3).

In order for the performance of this classifier to compete with that of the MLP, it is essential that all parameters are trained together, and with a discriminative objective function. Unsupervised discriminative training is also possible, using minimum classification error techniques [9]. However, in this article we take the simpler approach of training by supervised gradient descent. During training the softmax function is used to constrain the weights $w_{jk} = P(r_j, s_k)$ to lie in $[0, 1]$, and sum to one.

$$w_{jk} = e^{\alpha_{jk}} / \sum_{l,m} e^{\alpha_{lm}} \tag{5}$$

This gives the full set of parameters to be trained as $(\mu_{ij}, v_{ij}, \alpha_{jk})$, for $i = 1 \ldots n_x$, $j = 1 \ldots n_y$, $k = 1 \ldots n_z$.

3.1 Parameter initialisation

Any hill climbing procedure can encounter problems with local minima, so that system performance may be very sensitive to the initial parameter values used. In the context of the TIDigits connected digits ASR task, the following two methods were tested [13] for initialising the RBF layer parameters (means, variances, and priors $P(r_j)$):

- Randomly assign each data point to an RBF centre, followed by k-means clustering and likelihood maximisation by Expectation Maximisation (EM).
- Use HTK (version 1.5) [18] to train a set of 400 pooled Gaussians, using the Baum-Welch forward-backward training algorithm, with embedded realignment.

As well as training the RBF layer parameters, HTK also trains mix weights $P(r_j|s_k)$ for each of the hidden states as specified by whatever HMM structure is to be used in recognition. Whichever of the above methods was used, the trained HMM model was also used to provide a training data segmentation, from which we can estimate $P(s_k)$. Once the Gaussian parameters were initialised, two methods were tested for initialising the weights w, using the given segmentation:

- Use HMM trained mix weights $P(r_j|s_k)$ only:

$$w_{jk} = P(r_j, s_k) = P(r_j|s_k)P(s_k) \tag{6}$$

- Use HMM trained Gaussians only (see Appendix A for derivation of this rule):

$$P(r_j) = \sum_k P(r_j|s_k)P(s_k), \quad P(s_k|r_j) \cong \sum_{x_i \in s_k} y_j(x_i) / \sum_i y_j(x_i), \quad w_{jk} = P(r_j)P(s_k|r_j) \tag{7}$$

Of these different RBF layer and output layer initialisation methods, the best results *by far* were obtained using RBFs trained using HTK, and output weights trained using Eq.(7).

Before gradient descent training, the auxiliary parameters α were then initialised as:

$$\alpha_{jk} = \log(w_{jk}) \tag{8}$$

3.2 Error gradient calculation

Whichever error function E is used, the derivatives of E with respect to each of the model parameters were obtained by the usual "error back propagation" (EBP) approach, first calculating the "delta" values for each output unit. See Appendix B for details of EBP algorithm and derivation of Eqs(9 - 12):

$$\delta_k = \frac{\partial E}{\partial net_k} = \sum_l \frac{\partial E}{\partial z_l} \frac{(\delta_{kl} - z_l)}{p(x)} \tag{9}$$

$$\frac{\partial E}{\partial \mu_{ij}} = \frac{(x_i - \mu_{ij})}{v_{ij}} y_j \sum_k w_{jk} \delta_k \tag{10}$$

$$\frac{\partial E}{\partial v_{ij}} = \left(\frac{(x_i - \mu_{ij})^2}{v_{ij}} - 1 \right) \frac{y_j}{2 v_{ij}} \sum_k w_{jk} \delta_k \tag{11}$$

$$\frac{\partial E}{\partial \alpha_{jk}} = w_{jk}(1 - w_{jk}) y_j \delta_k \tag{12}$$

If τ_l is the target posterior for class l, then for three common error functions (to be minimised):

$$E = \sum_{i,k} (z_k(x_i) - t_k(x_i))^2 \quad \text{: mean square error} \tag{13}$$

$$E = -\sum_{i,k} t_k(x_i) \log z_k(x_i) \quad \text{: cross-entropy} \tag{14}$$

$$E = -\sum_{i,k} z_k(x_i) t_k(x_i) \quad \text{: correlation} \tag{15}$$

we have $\frac{\partial E}{\partial z_l} = \sum_i \frac{\partial E_i}{\partial z_l}$, with $\frac{\partial E_i}{\partial z_l}$ (dropping the i subscript) as:

$$z_l - \tau_l \quad \text{: mean square error} \tag{16}$$

$$-\tau_l / z_l \quad \text{: cross-entropy} \tag{17}$$

$$-\tau_l \quad \text{: correlation} \tag{18}$$

Best results here used the cross-entropy objective.

3.3 Gradient descent iteration

A constant "momentum" factor $\theta = 1$ is used, and an adaptable learning rate, φ [4]. With $g = \nabla E$, $g_0 = 0$, $dw_0 = 0$, $\varphi_0 = 1$, we have (where $\hat{g}$ is a unit vector):

$$\varphi_{t+1} = \varphi_t(1 - 0.5\hat{dw_t} \cdot \hat{g}_t) \tag{19}$$

$$dw_{t+1} = \varphi_{t+1}(\theta dw_t - \hat{g}_t) \tag{20}$$

$$w_{t+1} = w_t + dw_{t+1} \tag{21}$$

Training continues until the correct state classification rate on the cross-validation set stops increasing.

The gradient with respect to *all* IDCN parameters was evaluated, and all parameters updated, using all frames from a fixed number of utterances which were selected at random from the full training set. We found that very small samples led rapidly to one or more RBFs developing zero priors, from which they could not escape. As a compromise between processing speed and performance level at convergence, we settled on samples of 100 utterances.

It was found that further training of the RBF parameters by EM, after gradient descent training had converged, followed by application of Eq.(7), inevitably resulted in a very rapid increase in data likelihood, accompanied by an equally dramatic fall in classification accuracy. As a result this technique was not used.

4. Recognition with missing data

As outlined in Section 2, the way in which the IDCN is incorporated into a recognition system depends on whether or not the position of missing data is given. If it is given then expected posterior probabilities for each state need to be calculated just once, for the given position of missing data. Otherwise the expected posteriors need to be calculated for all possible positions of missing data, and averaged [3]. Whichever is the case, for any given position of missing data we may denote the present and missing components of the feature vector by (x_p, x_m). The estimate for $P(s_k|x)$ which results in the highest probability of correct classification is then given by the expected value of the classifier output function, conditioned on x_p and any knowledge κ_m which may constrain missing data values [10]. For the RBF classifier presented here this leads to the following estimates.

If nothing is known about the missing data then (see Appendix C for derivation of Eqs(22 - 24)):

$$\hat{z}_k(x) = E[P(s_k|x)|x_p] \propto \sum_j w_{jk} y_j(x_p) \tag{22}$$

If each missing feature has a limited range of possible values (as is the case for filterbank features, which are bounded below by zero and above by their observed value):

$$\hat{z}_k(x) = E[P(s_k|x)|x_p, x_m \in r_m] \tag{23}$$

$$\propto \sum_j w_{jk} y_j(x_p) \int_{r_m} y_j(x_m) dx_m \tag{24}$$

In Eqs. (22) and (24) $y_j(x_p)$ is the marginal diagonal Gaussian over the indicated x components. Posteriors $\hat{z}_k(x)$ are obtained by scaling the above values to sum to one across all classes.

It should be noted that it is only due to the consistent probabilistic interpretation of each stage of processing by this network that it is so simple to obtain the marginal posteriors in this way directly from the full system parameters.

5. Summary and conclusion

We have shown that an RBF network, in which the output layer uses Bayes' Rule to directly transform pooled mixture likelihoods from the RBF layer into a-posteriori class probabilities, is a suitable candidate network for classification with missing data. This is because it can be discriminatively trained, and the expected values of its posterior class probability outputs can readily be evaluated as a simple function of the original model parameters. We have further shown how this network can be incorporated into two different approaches to robust ASR. For the case where the position of missing data is known we can integrate the IDCN into an HMM system by replacing the usual state likelihoods by scaled state likelihoods output from the IDCN. In this case the posteriors based system should show some advantage over the likelihood based system due to discriminative training. However, ASR tests [13] have shown that severe problems arise with local minima during IDCN training by gradient descent, to the point that very little performance improvement is possible after parameter initialisation through normal non discriminative EM based HMM training. In fact, performance of the IDCN/HMM system was almost identical to that of a Gaussian mixture likelihood based HMM system, using the same missing feature theory and the same method for detecting missing data [16]. It is possible that the performance of the IDCN in this case could be improved by use of a more effective discriminative HMM training procedure, such as MCE [9] and/or boosting [15].

When the position of missing data is not known, the IDCN offers a new approach to multi-stream processing which should permit large numbers of feature streams to be combined with greatly reduced effort. This approach remains to be tested.

Acknowledgements

This work was supported by the EC/OFES (European Community / Swiss Federal Office for Education and Science) RESPITE project (REcognition of Speech by Partial Information TEchniques). Recognition tests for the methods presented in this report were carried out in collaboration with the speech group at Sheffield University, U.K.

Appendix A: Using HTK for both Gaussians and output layer weights initialisation

HTK can be used to estimate the Gaussian parameters μ_j, v_j, and mix weights $P(r_j|s_k)$ for each pooled Gaussian r_j and hidden state s_k. The trained HMMs can then be used to produce a state level segmentation. From this segmentation we can directly estimate state priors $P(s_k)$ from the relative frequency of occurrence of each state in the training data[1]. The number of free parameters to be trained should first be reduced by combining $P(r_j|s_k)$ and $P(s_k)$ into $P(r_j, s_k)$. We have tested two ways of doing this.

Method 1 uses Eq.(6), Section 3.1.

$$w_{jk} = P(r_j, s_k) = P(r_j|s_k)P(s_k)$$

Method 2 starts as method 1, but then estimates $w_{jk} = P(r_j, s_k) = P(r_j)P(s_k|r_j)$ by first estimating $P(r_j)$ using Eq.(7), and then estimating $P(s_k|r_j)$ using Eq.(7)

$$P(s_k|r_j) \cong \sum_{x_i \in s_k} y_j(x_i) / \sum_i y_j(x_i)$$

This is derived as follows:

$$P(s_k|r_j) = P(r_j, s_k)/P(r_j) = p(\bigcup_i x_i, r_j, s_k)/P(r_j)p(\bigcup_i x_i \Big| r_j, s_k) \tag{25}$$

$$= \sum_i p(x_i, r_j, s_k)/P(r_j) \sum_i p(x_i|r_j, s_k) \tag{26}$$

$$= \sum_i P(s_k|x_i, r_j)p(x_i, r_j)/P(r_j) \sum_i p(x_i|r_j) \tag{27}$$

$$= \sum_{x_i \in s_k} p(x_i|r_j) / \sum_i p(x_i|r_j) = \sum_{x_i \in s_k} y_j(x_i) / \sum_i y_j(x_i) \tag{28}$$

In the ASR tests made, it was found that method 2 gave far better recognition results. However, it is not clear why this should be so, and so this result may not generalise to other databases.

1. *If one or more states occur only a small number of times (so that the variance of the relative error in the relative frequency estimate is unacceptably high) then all state prior estimates should be weighted towards the uniform prior $1/n_k$. This is a commonly used probability estimate correction, which is directly related to the so called "m estimate", where the weighting factor is proportional to the "prior degree of belief" that the probabilities are all equal.*

Appendix B: Derivation of IDCN error gradient equations

Although in the present case the neural network under consideration has only one hidden layer, it is still helpful to make use of the "error back propagation" (EBP) theoretical framework, which is based around the idea that, for any network with connections between adjacent layers only, the contributions to the error gradient for parameters in the network at layer i can be obtained in terms of quantities δ_k which are first evaluated for each node in layer $(i+1)$.

$$z_l = \frac{net_k}{p(x)} = \frac{net_k}{\sum_m net_m}, \text{ where } net_k = \sum_j w_{jk} y_j(x) \tag{29}$$

The "delta rule" makes use of the chain rule for partial differentials, as follows:

$$\delta_k = \frac{\partial E}{\partial net_k} = \sum_l \frac{\partial E}{\partial z_l}\frac{\partial z_l}{\partial net_k} = \sum_l \frac{\partial E}{\partial z_l}\frac{\partial}{\partial net_k}\frac{net_l}{p(x)} = \sum_l \frac{\partial E}{\partial z_l}\frac{(\delta_{kl} - z_l)}{p(x)} \tag{30}$$

We can obtain the error gradient with respect to the output layer parameters w_{jk} as follows:

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial net_k}\frac{\partial net_k}{\partial w_{jk}} = \delta_k \frac{\partial}{\partial w_{jk}} \sum_l w_{lk} y_l(x) = \delta_k y_j(x) \tag{31}$$

$$\frac{\partial E}{\partial \alpha_{jk}} = \frac{\partial E}{\partial w_{jk}}\frac{\partial w_{jk}}{\partial \alpha_{jk}} \tag{32}$$

$$\frac{\partial w_{jk}}{\partial \alpha_{jk}} = \frac{\partial}{\partial \alpha_{jk}}\frac{e^{\alpha_{jk}}}{\sum_{l,m} e^{\alpha_{lm}}} = \left(e^{\alpha_{jk}} \sum_{l,m} e^{\alpha_{lm}} - e^{\alpha_{jk}} \sum_{l,m} \frac{\partial e^{\alpha_{lm}}}{\partial \alpha_{jk}} \right) \Big/ \left(\sum_{l,m} e^{\alpha_{lm}} \right)^2 \tag{33}$$

$$= e^{\alpha_{jk}} \left(\sum_{l,m} e^{\alpha_{lm}} - e^{\alpha_{jk}} \right) \Big/ \left(\sum_{l,m} e^{\alpha_{lm}} \right)^2 = w_{jk} - w_{jk}^2 = w_{jk}(1 - w_{jk}) \tag{34}$$

$$\frac{\partial E}{\partial \alpha_{jk}} = \delta_k y_j w_{jk}(1 - w_{jk}) \tag{35}$$

The error gradient for parameters μ_{ij} and v_{ij} in the hidden layer can be obtained in a similar way as follows:

$$\frac{\partial E}{\partial \mu_{ij}} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial \mu_{ij}}, \quad \frac{\partial E}{\partial y_j} = \sum_k \frac{\partial E}{\partial net_k}\frac{\partial net_k}{\partial y_j}, \quad \frac{\partial net_k}{\partial y_j} = \sum_l \frac{\partial}{\partial y_j} w_{lk} y_l = w_{jk} \tag{36}$$

$$\frac{\partial y_j}{\partial \mu_{ij}} = y_j \frac{(x_i - \mu_{ij})}{v_{ij}}, \quad \frac{\partial E}{\partial \mu_{ij}} = \frac{(x_i - \mu_{ij})}{v_{ij}} y_j \sum_k w_{jk} \delta_k \tag{37}$$

$$\frac{\partial y_j}{\partial v_{ij}} = \frac{y_j}{2v_{ij}} \left(\frac{(x_i - \mu_{ij})^2}{v_{ij}} - 1 \right), \quad \frac{\partial E}{\partial v_{ij}} = \frac{y_j}{2v_{ij}} \left(\frac{(x_i - \mu_{ij})^2}{v_{ij}} - 1 \right) \sum_k w_{jk} \delta_k \tag{38}$$

Appendix C: Derivation of expected class posterior probabilities

When a parametric classifier $z_k(x)$ is trained to estimate posterior class probabilities $P(q_k|x)$, and the position of missing components in the data vector x is given so that it can be partitioned into present and missing parts (x_p, x_m), then the estimate for the posterior probability for each class which is best in the sense that it gives the maximum probability of correct classification, is given by the expected value of the network output, conditioned on the data which is not missing, and any available knowledge κ_m which may be used to constrain the missing data values[1] [10].

$$\hat{z}_k(x) = E[z_k(x)|x_p, \kappa_m] \cong E[P(s_k|x)|x_p, \kappa_m] = \int_{R_m} \frac{p(s_k, x)}{p(x)} p(x_m|x_p, \kappa_m) dx_m \tag{39}$$

Here we will consider just two missing-data conditions. One in which nothing at all is known about the missing data values, and another in which the missing data is known to lie within a given range, R_m. In the second case we have:

$$p(x_m|x_p, \kappa_m) = \frac{p(x_m|x_p)}{\int_{R_m} p(x_m|x_p) dx_m} \text{ when } x \in R_m, \text{ else } = 0 \tag{40}$$

$$\text{so} \qquad \hat{z}_k(x) = \int_{R_m} \frac{p(s_k, x)}{p(x_p)p(x_m|x_p)} \frac{p(x_m|x_p)}{\int_{R_m} p(x_m|x_p) dx_m} dx_m = A \int_{R_m} p(s_k, x) dx_m \tag{41}$$

where A is independent of k. The integral can easily be evaluated as follows:

$$\int_{R_m} p(s_k, x) dx_m = \int_{R_m} \sum_j w_{jk} y_j(x) dx_m = \sum_j w_{jk} N_j(x_p) \int_{R_m} N_j(x_m|x_p) dx_m \tag{42}$$

Here we consider only the case of diagonal covariance, so that $N(x_m|x_p) = N(x_m)$ [14], and

$$\hat{z}_k(x) = A \sum_j w_{jk} N_j(x_p) \int_{R_m} N_j(x_m) dx_m \tag{43}$$

The integral in Eq.(43) can easily be evaluated as the product of univariate Gaussian integrals, each of which can be evaluated using the C standard erf function. If the missing data is unbounded then the integral is just unity and can be ignored. As $\sum_k P(s_k|x, \kappa_m) = 1$, the constant A can be eliminated, to obtain $\hat{z}_k(x)$ as follows:

$$\theta_k(x) = \sum_j w_{jk} N_j(x_p) \int_{R_m} N_j(x_m) dx_m \tag{44}$$

$$\hat{z}_k(x) = \theta_k(x) / \sum_l \theta_l(x) \tag{45}$$

---

1. *Note that the analysis presented in [1] regarding posteriors estimation with missing data is incorrect.*

References

[1] Ahmed, S. & Tresp, V. (1993) “Some solutions to the missing feature problem in vision”, in Advances in Neural Information Processing Systems 5, Morgan Kauffman, San Mateo, pp. 393-400.

[2] Bishop, C. (1995) Neural Networks for Pattern Recognition, Clarendon Press, Oxford.

[3] Bourlard, H. & Morgan, N. (1993) Connectionist Speech Recognition, Kluwer Academic Publishers, Boston.

[4] Chan, L.W. & Fallside, F. (1987) “An adaptive training algorithm for back-propagation networks”, Computer Speech and Language 2, pp.205-218.

[5] Hagen, A. & Morris, A.C. (in press) “Comparison of HMM experts with MLP experts in the full combination multi-band approach to robust ASR”, Proc. ICSLP-2000.

[6] Hagen, A., Morris, A.C. & Bourlard, H. (1998) “Sub-band based speech recognition in noisy conditions: The Full-Combination approach”, Research Report IDIAP-RR 98-15.

[7] Hermansky, H., Ellis, D. & Sharma, S. (2000) “Tandem connectionist feature stream extraction for conventional HMM systems”, Proc. ICASSP-2000, pp.1635-1638..

[8] Lippmann, R. P. & Carlson, B. A. (1997) “Using missing feature theory to actively select features for robust speech recognition with interruptions, filtering and noise”, Proc. Eurospeech’97, pp. 37-40

[9] McDermott, E. & Katagiri, S. (1994) “Prototype-based minimum classification error / generalised probabilistic descent training for various speech units”, Computer Speech and Language, No.8, pp.351-368.

[10] Morris, A.C., Cooke, M. & Green, P. (1998) “Some solutions to the missing feature problem in data classification, with application to noise robust ASR”, Proc. ICASSP'98, pp.737-740.

[11] Morris, A.C., Hagen, A. & Bourlard, H. (1999) “The full-combination subbands approach to noise robust HMM/ANN based ASR”, Proc. Eurospeech’99, pp.599-602.

[12] Morris, A.C., Hagen, A., Glotin, H. & Bourlard, H. (in press) “Multi-stream adaptive evidence combination for noise robust ASR”, Speech Communication.

[13] Morris, A.C., Josifovski, L., Bourlard, H., Cooke, M. & Green, P. (in press) “A neural network for classification with incomplete data: application to robust ASR”, Proc. ICSLP 2000.

[14] Morrison, D.F. (1990) Multivariate statistical methods, 3rd Edition, McGraw-Hill.

[15] Schwenk, H. (1999) “Using boosting to improve a hybrid HMM/neural network speech recogniser”, Proc. ICASSP’99. pp.1009-1012.

[16] Vizinho, A., Green, P., Cooke, M. & Josifovski, L. (1999) “Missing data theory, spectral subtraction and signal-to-noise estimation for robust ASR: An integrated study”, Proc. Eurospeech’99, pp.2407-2410.

[17] White, H. (1989) “Learning in artificial neural networks: A statistical perspective”, Neural Computing, 1, pp.425-464.

[18] Young, S.J. & Woodland, P.C. (1993) HTK Version 1.5: User, Reference and Programmer Manual, Cambridge University Engineering Dept., Speech Group.