Product of Gaussians for Speech Recognition

M.J.F. Gales and S.S. Airey

Cambridge University Engineering Department, Trumpington Street, Cambridge, CB2 1PZ, England

Abstract

Recently there has been interest in the use of classifiers based on the product of experts (PoE) framework. PoEs offer an alternative to the standard mixture of experts (MoE) framework. It may be viewed as examining the intersection of a series of experts, rather than the union as in the MoE framework. This paper presents a particular implementation of PoEs, the normalised product of Gaussians (PoG). Here each expert is a Gaussian mixture model. In this work, the PoG model is presented within a hidden Markov model framework. This allows the classification of variable length data, such as speech data. Training and initialisation procedures are described for this PoG system. The relationship of the PoG system with other schemes, including covariance modeling schemes, is also discussed. In addition the scheme is shown to be related to a standard speech recognition approach, multiple stream systems. The PoG system performance is examined on an automatic speech recognition task, Switchboard. The performance is compared to standard Gaussian mixture systems and multiple stream systems.

1 Introduction

Mixture of Gaussians (MoG) are commonly used as the state representation in hidden Markov model (HMM) based speech recognition. These Gaussian mixture models are easy to train using expectation maximisation (EM) techniques (4) and are able to approximate any distribution given a sufficient number of components. However, only a limited number of parameters can be effectively trained given a finite quantity of training data. This limitation restricts the ability of MoG systems to model highly complex distributions. A range of *distributed* representations have been developed to overcome this

Email addresses: mjfg@eng.cam.ac.uk (M.J.F. Gales), ssa26@eng.cam.ac.uk (S.S. Airey).

URL: http://www-svr.eng.cam.ac.uk/~mjfg (M.J.F. Gales).

problem. These distributed representations may be split into two basic forms. The first assumes that the sources are *asynchronous*. The second assumes that the sources are *synchronous*.

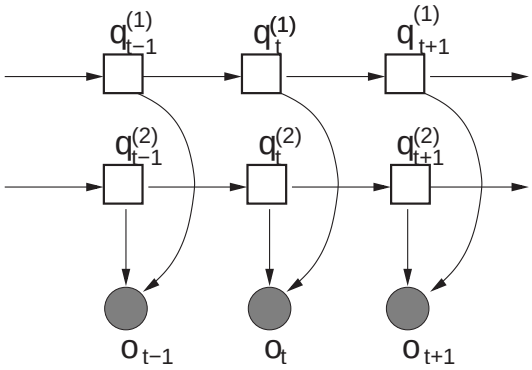

Fig. 1. 2-stream asynchronous distributed representation. Observed values are shaded, unobserved values are unshaded. Circles are used to represent continuous values, squares discrete values. The absence of an arrow indicates independence.

A dynamic Bayesian network (DBN) for an asynchronous representation is shown in figure 1. In this representation, there are multiple sources that change "state" at different time instances. The observations are assumed to be conditionally independent given the states of the underlying sources. Factorial HMMs (8; 6) are a standard example of an asynchronous distributed system. In practice, this form has limited use because separate streams are seldomly completely asynchronous. Some constraints between the streams are usually imposed. Models of this form include loosely coupled HMMs (12) and the mixed memory model (17).

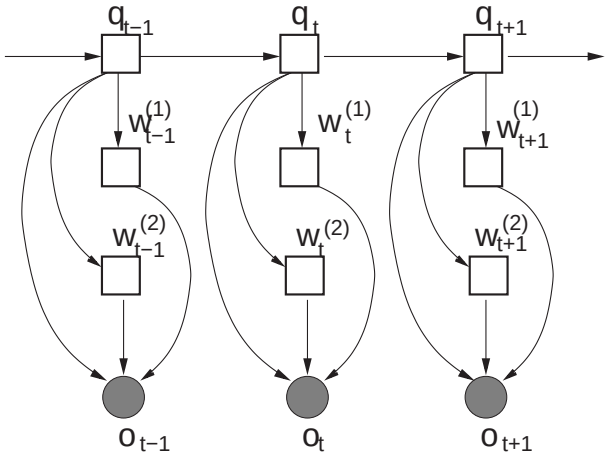

Fig. 2. 2-stream synchronous representation with mixture models

An alternative to the factorial HMM scheme is to use a single state process and have a distributed form associated with the state distribution. A standard approach is to use a mixture model for each source. The mixture models from each source change at the same time instance. Hence it is a *synchronous* representation. The DBN for this representation is shown in figure 2. The

distributions for the two underlying streams are mixture models, the generative component at each time instance, t, is given by $w_t^{(s)}$ for stream s. A standard example of this form of representation is the multiple stream system implemented in HTK (20).

Irrespective of whether the underlying processes are synchronous or asynchronous, an important consideration is how the individual streams combine to yield the observation, or observation sequence, distribution. A variety of combination schemes have been proposed including discrete stream systems (21), linear stream transformations (6), parallel model combination (7) and independent streams (2). A brief overview of these schemes is given in (6). All these schemes attempt to find the appropriate balance between the number of model parameters and assumptions of stream independence.

The distributed representation investigated in this paper is the products of experts (PoE) (10) framework using Gaussian mixture models (GMMs) as the experts. The general PoE framework cannot be viewed as a distributed representation. However, since the product of two Gaussian distributions is itself Gaussian distributed, the special case of PoE that uses Gaussian experts is a distributed representation. In the PoE framework, the likelihoods from all the experts are multiplied together to form the system output. The PoE system output can be thought of as an *intersection* of all the individual experts. In contrast, the output from the standard mixture of experts (MoE) system, of which the MoG is one example, is a *union* of all the individual experts. For a MoE system, $\mathcal{M}$, composed of S experts, the output likelihood may be expressed as

$$p(\mathbf{o}_t|\mathcal{M}) = \sum_{s=1}^{S} c^{(s)} p(\mathbf{o}_t|\mathcal{M}^{(s)}) \tag{1}$$

where $c^{(s)}$ is the prior for expert $\mathcal{M}^{(s)}$. For this to be a valid PDF, $\sum_{s=1}^{S} c^{(s)} = 1$. The equivalent output likelihood for a PoE system may be expressed as

$$p(\mathbf{o}_t|\mathcal{M}) = \frac{1}{Z} \prod_{s=1}^{S} p(\mathbf{o}_t|\mathcal{M}^{(s)}) \tag{2}$$

$$Z = \int_{\mathcal{R}^d} \prod_{s=1}^{S} p(\mathbf{o}|\mathcal{M}^{(s)}) d\mathbf{o}. \tag{3}$$

where the integral is over the d-dimensional feature-space. Z is the normalisation term required to yield a valid PDF. The PoE framework may be applied in both the synchronous and asynchronous systems.

Training MoE systems using the EM algorithm is usually straight-forward. However, training PoEs is significantly more complicated, largely as a result

of the normalisation term. This complexity has motivated various approximate training schemes for the PoE framework (10). PoEs have previously been applied to time varying signals (3). Discrete HMMs were used in an asynchronous framework to classify character strings . This paper investigates modeling the states of a HMM in a synchronous framework using PoE systems in which the individual experts are Gaussian or MoG. Using this form of expert, the training is dramatically simplified compared to the general PoE case. In addition, this form of model is, under certain restrictions, related to multiple stream systems.

An interesting aspect of distributed representations of observations, including the PoE framework described here, is the nature of the contributing stream distributions. For statistical pattern processing an overriding constraint is that the final non-distributed observation, or observation sequence, probability distribution be valid. It must satisfy the following constraints:

$$p(\mathbf{o}|\mathcal{M}) \geq 0; \qquad \int_{\mathcal{R}^d} p(\mathbf{o}|\mathcal{M}) d\mathbf{o} = 1. \tag{4}$$

However, the individual streams do not necessarily have to satisfy these constraints. One such example is the extended maximum likelihood linear transform covariance representation (13), where there are "negative" inverse variances. The effects of this relaxation of the constraints on the experts is investigated for the case of product of Gaussians and MoG.

This paper is organised as follows. The next section describes the general product of Gaussians and related schemes. Section 3 discusses the product of Gaussians as a distributed representation, examining the effects of relaxing the constraint that the individual experts be valid probability distributions. The training of the system is then described. In section 5, the relationship between the product of Gaussian system and the multiple stream system is discussed. Results on a large vocabulary speech recognition task are then described in section 6.

2 Product of Gaussians Systems

This section details different forms of the product of Gaussians model. The PoG pancake (PoGP) model, described in section section 2.2, has previously been investigated in detail (19). The PoGP model is described within the framework of covariance matrix modeling and is related to probabilistic PCA (18) and factor analysis (16). In contrast, this paper concentrates on the attributes of PoG systems within a mixture model and HMM framework. Two new forms of representation are discussed. The first uses MoG as the experts. The second form, and the one evaluated here, considers normalised versions of the product

of individual components from the MoG experts. These forms, in addition to the PoGP model, are described below.

2.1 General Product of Gaussians

The product of two Gaussian likelihoods is itself Gaussian distributed when appropriately normalised. For the product of S multivariate Gaussian distributions, this may be written as

$$\begin{aligned} p(\mathbf{o}|\mathcal{M}) &= \frac{1}{Z}\prod_{s=1}^{S}\mathcal{N}(\mathbf{o};\boldsymbol{\mu}^{(s)},\boldsymbol{\Sigma}^{(s)}) \\ &= \mathcal{N}(\mathbf{o};\boldsymbol{\mu},\boldsymbol{\Sigma}) \end{aligned} \tag{5}$$

where Z is the normalisation term. The mean and the covariance matrix of the resulting distribution are

$$\boldsymbol{\mu} = \boldsymbol{\Sigma}\left(\sum_{s=1}^{S}\boldsymbol{\Sigma}^{(s)-1}\boldsymbol{\mu}^{(s)}\right) \tag{6}$$

$$\boldsymbol{\Sigma} = \left(\sum_{s=1}^{S}\boldsymbol{\Sigma}^{(s)-1}\right)^{-1} \tag{7}$$

In contrast to many PoE models, there is a simple expression for the normalisation term, Z. From equation 5, the normalisation term may be shown to be

$$Z = \frac{(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}}{\prod_{s=1}^{S}(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma}^{(s)}|^{\frac{1}{2}}}\exp\left[\frac{1}{2}\left(\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - \sum_{s=1}^{S}(\boldsymbol{\mu}^{(s)\prime}\boldsymbol{\Sigma}^{(s)-1}\boldsymbol{\mu}^{(s)})\right)\right] \tag{8}$$

where d is the dimensionality of the feature vector. If the covariance matrices are full then the product of Gaussians does not add additional modeling power. However, in models that use constrained forms of the covariance matrix, such as the PoGP described in the next section, then the simple product of Gaussians allows more complex covariance matrix modeling.

2.2 Product of Gaussian Pancakes

Since the product of a set of Gaussians is itself a Gaussian distribution, if single component GMMs are used for each expert, the PoE framework will simply allow an alternative version of covariance modeling. One form of this

covariance model is the product of Gaussian pancake (GP). In this model for GP s the probability contour is assumed to be equally stretched in directions $\mathbf{v}_1^{(s)}, \ldots, \mathbf{v}_{d-1}^{(s)}$ and contracted in direction $\mathbf{w}^{(s)}$, such that $\mathbf{v}_1^{(s)}, \ldots, \mathbf{v}_{d-1}^{(s)}, \mathbf{w}^{(s)}$ form a $d \times d$ matrix of the normalised eigenvectors of the covariance matrix $\mathbf{\Sigma}^{(s)}$. The inverse covariance matrix of a particular d-dimensional Gaussian pancake, s, is then

$$\begin{aligned}\mathbf{\Sigma}^{(s)-1} &= \sum_{i=1}^{d-1} \lambda_0^{(s)} \mathbf{v}_i^{(s)} \mathbf{v}_i^{(s)\prime} + \lambda^{(s)} \mathbf{w}^{(s)} \mathbf{w}^{(s)\prime} \\ &= \lambda_0^{(s)} \mathbf{I} + \mathbf{a}^{(s)} \mathbf{a}^{(s)\prime}\end{aligned} \tag{9}$$

where $\mathbf{a}^{(s)} = \mathbf{w}^{(s)} \sqrt{\lambda^{(s)} - \lambda_0^{(s)}}$ and $\lambda^{(s)}$ and $\lambda_0^{(s)}$ are the inverse variances in the directions of contraction and elongation respectively. If a series of S of these pancakes are multiplied together then the final resultant inverse covariance matrix, $\mathbf{\Sigma}^{-1}$, is given by

$$\begin{aligned}\mathbf{\Sigma}^{-1} &= \sum_{s=1}^{S} \left(\lambda_0^{(s)} \mathbf{I} + \mathbf{a}^{(s)} \mathbf{a}^{(s)\prime} \right) \\ &= \lambda \mathbf{I} + \mathbf{A}\mathbf{A}'\end{aligned} \tag{10}$$

where $\lambda = \sum_{s=1}^{S} \lambda_0^{(s)}$ and $\mathbf{A} = \left[\mathbf{a}^{(1)} \ \ldots \ \mathbf{a}^{(S)} \right]$. This is closely related to factor analysis (19), but each "factor" now relates to the inverse covariance matrix. It is interesting that the form shown for PoGP in equation 10 is very similar to the form of extended maximum likelihood linear transform (EMLLT) (13). This relationship is examined in more detail in section 3.

2.3 Product of Mixtures of Gaussians

In HMM-based automatic speech recognition systems it is common to use MoGs to model the state distribution. A PoE framework with Gaussian mixture model experts may be used as an alternative state distribution model. The product of mixtures of Gaussians (PoMoG) is a version of synchronous distributed representation. The distribution is obtained by multiplying the likelihoods from the individual streams, where each stream is represented by a MoG expert. In this case, equation 2 may be expressed as

$$p(\mathbf{o}_t|\mathcal{M}) = \frac{1}{Z} \prod_{s=1}^{S} \left(\sum_{m=1}^{M^{(s)}} c_m^{(s)} \mathcal{N}\left(\mathbf{o}_t; \boldsymbol{\mu}_m^{(s)}, \mathbf{\Sigma}_m^{(s)} \right) \right) \tag{11}$$

where $M^{(s)}$, $c_m^{(s)}$, $\boldsymbol{\mu}_m^{(s)}$, and $\boldsymbol{\Sigma}_m^{(s)}$ denote the number of components in stream s, the prior, mean and covariance matrix of component m of stream s. Since the product of two or more Gaussians is itself Gaussian distributed, the product of the mixture of Gaussians may be expressed as a mixture model in the *product space*. Equation 11 may be rewritten as

$$p(\mathbf{o}_t|\mathcal{M}) = \frac{1}{Z}\sum_{m_1=1}^{M^{(1)}}\cdots\sum_{m_S=1}^{M^{(S)}}\prod_{s=1}^{S} c_{m_s}^{(s)}\mathcal{N}\left(\mathbf{o}_t;\boldsymbol{\mu}_{m_s}^{(s)},\boldsymbol{\Sigma}_{m_s}^{(s)}\right) \tag{12}$$

$$= \frac{1}{Z}\sum_{m_1=1}^{M^{(1)}}\cdots\sum_{m_S=1}^{M^{(S)}} c_{\mathbf{m}}K_{\mathbf{m}}\mathcal{N}(\mathbf{o}_t;\boldsymbol{\mu}_{\mathbf{m}},\boldsymbol{\Sigma}_{\mathbf{m}})$$

$$= \frac{1}{Z}\sum_{\mathbf{m}} c_{\mathbf{m}}K_{\mathbf{m}}\mathcal{N}(\mathbf{o}_t;\boldsymbol{\mu}_{\mathbf{m}},\boldsymbol{\Sigma}_{\mathbf{m}}) \tag{13}$$

where $\mathbf{m} = \begin{bmatrix} m_1 \ \ldots \ m_S \end{bmatrix}'$ determines a particular *meta-component* and m_s specifies the component from stream s. $K_{\mathbf{m}}$ is an observation-independent normalisation and can be expressed as

$$K_{\mathbf{m}} = \frac{(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma}_{\mathbf{m}}|^{\frac{1}{2}}}{\prod_{s=1}^{S}(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma}_{m_s}^{(s)}|^{\frac{1}{2}}}\exp\left[\frac{1}{2}\left(\boldsymbol{\mu}_{\mathbf{m}}'\boldsymbol{\Sigma}_{\mathbf{m}}^{-1}\boldsymbol{\mu}_{\mathbf{m}} - \sum_{s=1}^{S}(\boldsymbol{\mu}_{m_s}^{(s)\prime}\boldsymbol{\Sigma}_{m_s}^{(s)-1}\boldsymbol{\mu}_{m_s}^{(s)})\right)\right] \tag{14}$$

The summation in this form is over all meta-component combinations, where a meta-component is the product of one component from each MoG expert. The mean, covariance matrix and prior of each meta-component $\mathbf{m}$ may be expressed as

$$\boldsymbol{\mu}_{\mathbf{m}} = \boldsymbol{\Sigma}_{\mathbf{m}}\left(\sum_{s=1}^{S}\boldsymbol{\Sigma}_{m_s}^{(s)-1}\boldsymbol{\mu}_{m_s}^{(s)}\right) \tag{15}$$

$$\boldsymbol{\Sigma}_{\mathbf{m}} = \left(\sum_{s=1}^{S}\boldsymbol{\Sigma}_{m_s}^{(s)-1}\right)^{-1} \tag{16}$$

$$c_{\mathbf{m}} = \prod_{s=1}^{S} c_{m_s}^{(s)}. \tag{17}$$

In this form, the effective number of components, M, in the PoE model is the number of possible combinations of components from each MoG,

$$M = \prod_{s=1}^{S} M^{(s)} \tag{18}$$

The normalisation term for PoMoG can be written as

$$Z = \sum_{\mathbf{m}} c_{\mathbf{m}} K_{\mathbf{m}}. \tag{19}$$

In contrast to many other forms of PoE systems, the normalisation term has a simple analytical form. This allows standard gradient descent optimisation schemes to be used to find the model parameters.

2.4 Normalised Product of Gaussians

An alternative form of product of Gaussians (PoG) model normalises each meta-component $\mathbf{m}$ rather than normalising at the product of MoG level. This is the normalised PoG system examined in this paper. As PoG will be used to model the state distributions for an HMM system, the likelihoods will now be conditioned on the state of the model (the dependence on the model parameters will be implicit). The likelihood for state q_t may be written as

$$\begin{aligned} p(\mathbf{o}_t|q_t) &= \sum_{\mathbf{m}} \frac{c_{\mathbf{m}}}{K_{\mathbf{m}}} \prod_{s=1}^{S} \mathcal{N}\left(\mathbf{o}_t; \boldsymbol{\mu}_{m_s}^{(s)}, \boldsymbol{\Sigma}_{m_s}^{(s)}\right) \\ &= \sum_{\mathbf{m}} c_{\mathbf{m}} \mathcal{N}\left(\mathbf{o}_t; \boldsymbol{\mu}_{\mathbf{m}}, \boldsymbol{\Sigma}_{\mathbf{m}}\right) \end{aligned} \tag{20}$$

where $K_{\mathbf{m}}$ is the normalisation term for the meta-component given in equation 13. By using this form of normalisation it possible to closely relate this form of model to multiple stream systems, as detailed in 5.

3 PoGs as a Distributed Representation

In the previous section the product of Gaussians model and related schemes were described. This section examines the product of GMM experts and describes the additional flexibility introduced if the standard PDF constraints on the individual stream experts are relaxed. Three forms of constraints are considered. The first affects the component priors, the other two are constraints of the covariance matrix.

3.1 Component Priors

For each of the experts to be a valid PDF the component priors for each expert must satisfy

$$\sum_{m=1}^{M^{(s)}} c_m^{(s)} = 1; \quad c_m^{(s)} \geq 0 \tag{21}$$

These are then combined together using equation 17 to yield $c_{\mathbf{m}}$. Strictly, however, the priors are only required to be valid for the final distribution in the product space. In this case, the constraints on component priors are relaxed so now the constraint is

$$\sum_{\mathbf{m}} c_{\mathbf{m}} = 1; \quad c_{\mathbf{m}} \geq 0. \tag{22}$$

In itself, this relaxation of the constraints does not alter the expert prior estimation. However, it is interesting to consider a model where the priors for each meta-component are explicitly estimated from the training data rather than obtained from a product of the expert priors. From equation 20, the ML estimation of meta-component priors is a standard problem of estimating component priors in a mixture model. However here meta-components, rather than individual expert components, are considered.

For the case of the PoMoG system this is more interesting. In the PoMoG system the normalisation term for each meta-component $K_{\mathbf{m}}$ and Z must also be taken in to account. From equation 13 the likelihood may be expressed as

$$p(\mathbf{o}_t|\mathcal{M}) = \sum_{\mathbf{m}} \left(\frac{w_{\mathbf{m}} K_{\mathbf{m}}}{\sum_{\mathbf{m}} w_{\mathbf{m}} K_{\mathbf{m}}} \right) \mathcal{N}(\mathbf{o}_t; \boldsymbol{\mu}_{\mathbf{m}}, \boldsymbol{\Sigma}_{\mathbf{m}}) \tag{23}$$

where $w_{\mathbf{m}}$ is the "weight" for meta-component $\mathbf{m}$ to denote that it is no longer required to be a valid prior. Now for the priors to satisfy 22

$$w_{\mathbf{m}} K_{\mathbf{m}} \geq 0 \tag{24}$$

with the additional constraint that at least one of the meta-component values is greater than zero. Since $w_{\mathbf{m}}$ can take any value, the effect of $K_{\mathbf{m}}$ can be "undone" by the appropriate value of $w_{\mathbf{m}}$. Thus, letting

$$c_{\mathbf{m}} = \frac{w_{\mathbf{m}} K_{\mathbf{m}}}{\sum_{\mathbf{m}} w_{\mathbf{m}} K_{\mathbf{m}}} \tag{25}$$

it is possible to directly estimate the meta-component prior, $c_{\mathbf{m}}$, with the standard prior constraints. In this case, the PoMoG system is identical to the PoG system. Product of HMMs may be treated in a similar fashion. This is discussed in detail in (5).

Meta-component priors estimated directly as in equation 25 will be referred to as *ML meta-component priors*. In contrast, meta-component priors calculated

from the product of expert priors as in equation 17 will be referred to as *product meta-component priors.*

3.2 Covariance Matrices

For the covariance matrices to yield valid distributions for each expert, $\mathbf{\Sigma}_{m_s}^{(s)}$ must be positive definite for all components of all experts. In a similar fashion to the priors, this constraint may be relaxed so that only the final covariance matrix $\mathbf{\Sigma_m}$ need be positive definite. The meta-component variance is found from the individual experts using

$$\mathbf{\Sigma_m} = \left(\sum_{s=1}^{S} \mathbf{\Lambda}_{m_s}^{(s)} \right)^{-1} \tag{26}$$

where $\mathbf{\Lambda}_{m_s}^{(s)}$ is used, rather than the inverse covariance, to illustrate that the individual covariance matrices need not be valid. There are two attributes of the covariance that may be relaxed. First, the individual expert matrices need not be of full rank. Second, the elements on the leading diagonal need not be positive.

- **Non-full rank matrices**: each of the individual expert covariance matrices are not required to be of full rank provided that the final covariance matrix is of full rank. One extreme scheme that clearly illustrates this is the EMLLT model (13). Here the inverse covariance matrix is formed by combining a set of experts each of whose inverse covariance matrices are rank-1. This method may be viewed as generating a set of experts by projecting into a series of one-dimensional spaces.

$$\begin{aligned} \mathbf{\Sigma}_{\mathbf{m}}^{-1} &= \mathbf{A} \mathbf{\Lambda_m} \mathbf{A}' \\ &= \sum_{s=1}^{S} \lambda_{m_s}^{(s)} \mathbf{a}^{(s)} \mathbf{a}^{(s)\prime} \end{aligned} \tag{27}$$

  where $\mathbf{A}$ is a $d \times S$ matrix, $S \geq d$, and $\mathbf{\Lambda_m}$ is a diagonal, $S \times S$, matrix with the s^{th} element on the leading diagonal being $\lambda_{m_s}^{(s)}$. $\lambda_{m_s}^{(s)}$ may be interpreted as the inverse variance in the s^{th} projected dimension. The EMLLT scheme is related to the PoGP model. This relationship becomes clear if an additional expert of full rank is used as well. Let [1]

$$p(\mathbf{o}_t | \mathcal{M}^{(S+1)}) = \mathcal{N}(\mathbf{o}_t; \mathbf{0}, \frac{1}{\lambda}\mathbf{I}). \tag{28}$$

[1] A more general covariance matrix, $\mathbf{\Sigma}^{(w)}$, may be used. Here an identity matrix is used to illustrate the relationship with PoGP. For the EMLLT system ML-estimates of each meta-components means is used rather the form determined by equation 15.

The product of all $S+1$ experts yields a Gaussian with covariance matrix

$$\boldsymbol{\Sigma}_{\mathbf{m}}^{-1} = \sum_{s=1}^{S} \left(\lambda_0^{(s)} \mathbf{I} + \lambda_{m_s}^{(s)} \mathbf{a}^{(s)} \mathbf{a}^{(s)\prime} \right). \tag{29}$$

where again $\lambda = \sum_{s=1}^{S} \lambda_0^{(s)}$. The additional expert is used by the PoGP model to ensure that all the individual experts are valid distributions, whereas the EMLLT process would simply view it as additional rows of the transformation matrix.

Though the two schemes have a similar form, there are some fundamental differences. The first is that EMLLT is designed for covariance matrix modeling where the matrix $\mathbf{A}$ is shared over multiple components. In contrast, PoGP assumes only a single Gaussian. For PoGP systems, a value of $S > d$ will simply yield a full-covariance matrix. For EMLLT, if the form used in equation 27 is used then $d \leq S \leq \frac{d}{2}(d+1)$.When the form of equation 29 is used then $S \leq \frac{d}{2}(d+1)$.

- **"Negative" variances**: the leading diagonal of a valid covariance matrix must be non-negative. However, this requirement may be relaxed for the individual experts in a PoE framework. For the PoG system, the meta-component covariance matrix is given by

$$\boldsymbol{\Sigma}_{\mathbf{m}} = \left(\sum_{s=1}^{S} \boldsymbol{\Lambda}_{m_s}^{(s)} \right)^{-1}. \tag{30}$$

Provided the elements on the leading diagonal are positive in the final meta-component variance, $\boldsymbol{\Sigma}_{\mathbf{m}}$, the terms from the experts, $\boldsymbol{\Lambda}_{m_s}^{(s)}$, need not be positive. Thus an unconstrained optimisation scheme may be used.

PoGs are not the only form of model in which negative variances can occur. The EMLLT framework (13) also affords this flexibility, as does the generalised linear Gaussian model framework (15; 14) when the size of the state-space is greater than the observation space. In (13) it is shown that having negative values of $\lambda_{m_s}^{(s)}$, the "variance" of the state distribution in dimension s, yields improved training data likelihood and classification results for the EMLLT model.

For the PoG scheme investigated in this paper, only the negative variance element is relevant as the experts in all cases are of the same dimension as the observation space and the matrices are constrained to be diagonal.

3.3 Likelihood Calculation

The computational cost of decoding is an important factor in deciding which form of representation to use. For the standard PoE system decoding is relatively cheap, as it is only necessary to compute the individual stream log-

likelihoods and then combine them. This is significantly less computationally expensive than computing the likelihood directly from the meta-components in the product space. In contrast, computing the log-likelihoods for the distributed representation is not so straightforward.

In the distributed representation, the calculation of the log contribution for each stream may be split into two distinct parts. Since each expert is not required to be a valid PDF, the contribution will be denoted as $f(\mathbf{o}_t, m)$ and the normalisation terms will be ignored for the time being:

$$f(\mathbf{o}_t, m_s) = -\frac{1}{2}(\mathbf{o}_t - \boldsymbol{\mu}_{m_s}^{(s)})' \boldsymbol{\Lambda}_{m_s}^{(s)} (\mathbf{o}_t - \boldsymbol{\mu}_{m_s}^{(s)}). \tag{31}$$

The only requirement to compute $f(\mathbf{o}_t, m_s)$ is the existence of $\boldsymbol{\Lambda}_{m_s}^{(s)}$, rather than requiring the inverse of a covariance matrix to exist. It is then possible to write the meta-component likelihood as

$$p(\mathbf{o}_t|\mathbf{m}) = \frac{1}{(2\pi)^{\frac{d}{2}} |\boldsymbol{\Sigma}_{\mathbf{m}}|^{\frac{1}{2}}} \exp\left(\sum_{s=1}^{S} f(\mathbf{o}_t, m_s) \right). \tag{32}$$

The likelihood of a state generating the observation may then be expressed as

$$p(\mathbf{o}_t|\mathcal{M}) = \sum_{\mathbf{m}} c_{\mathbf{m}} p(\mathbf{o}_t|\mathbf{m}). \tag{33}$$

For efficiency, the meta-component prior can be combined with the meta-component normalisation term. The efficiency of this style of approach for non-full-rank matrices is described in (13).

4 Training PoG Systems

The maximum likelihood (ML) PoG system training is more complicated than that for a standard HMM or multiple stream system. To estimate the product space means and variances, a generalised EM formulation is used. The complete data set for the auxiliary function is based on the observations and the posterior probability at time t of a meta-component $\mathbf{m}$, given the current model parameters and the all the observations, $\gamma_t^{(\mathbf{m})}$. A gradient descent scheme is used to maximise the auxiliary function as there are no closed form solutions to optimise the means and variances. For a PoG system the auxiliary function may be written as

$$\mathcal{Q}(\mathcal{M}, \hat{\mathcal{M}}) = \sum_{t=1}^{T} \sum_{\mathbf{m}} \gamma_t^{(\mathbf{m})} \log\left(c_{\mathbf{m}}\right) + \sum_{t=1}^{T} \sum_{\mathbf{m}} \gamma_t^{(\mathbf{m})} \log\left(\mathcal{N}(\mathbf{o}_t; \boldsymbol{\mu}_{\mathbf{m}}, \boldsymbol{\Sigma}_{\mathbf{m}})\right) \quad (34)$$

$$= \mathcal{Q}^{(c)}(\mathcal{M}, \hat{\mathcal{M}}) + \mathcal{Q}^{(\mathcal{N})}(\mathcal{M}, \hat{\mathcal{M}}) \quad (35)$$

Initially, only the update of the product space Gaussian distributions is considered. Expressing the auxiliary function in terms of the individual stream distributions and using the fact that the individual experts have diagonal covariance matrices yields

$$\mathcal{Q}^{(\mathcal{N})}(\mathcal{M}, \hat{\mathcal{M}}) = \sum_{t=1}^{T} \sum_{\mathbf{m}} \sum_{i=1}^{d} \frac{\gamma_t^{(\mathbf{m})}}{2} \left(\log\left(\sum_{s=1}^{S} \frac{1}{\sigma_{m_s i}^{(s)2}} \right) - (o_{ti}\mu_{\mathbf{m}i})^2 \left(\sum_{s=1}^{S} \frac{1}{\sigma_{m_s i}^{(s)2}} \right) \right). \quad (36)$$

There are no simple closed-form solutions to finding the model parameters, so gradient descent schemes are used. Each dimension of the system may be optimised separately so it is feasible to consider second-order techniques. For systems of reasonable size the Hessian may be explicitly computed. For further details of this training see (1). For the scheme implemented, statistics are stored for each Gaussian component in the product space. It is then possible to guarantee that the auxiliary function increases at each step. Two forms of the optimisation may be used depending on whether a distributed representation is being used.

- **Unconstrained variances**: the form in equation 36 may be directly used.
- **Positive variances**: rather than optimising $\sigma_{m_s i}^{(s)2}$, $\log(\sigma_{m_s i}^{(s)2})$ may be used instead. Now it is only necessary to ensure that $\log(\sigma_{m_s i}^{(s)2})$ is real to guarantee that $\sigma_{m_s i}^{(s)2}$ is positive [2].

Storing statistics for components in product space makes training systems with large numbers of streams, or components per stream, impractical. Alternatively, it is possible to use more general gradient descent learning schemes where it is necessary to only store updates with the individual stream components. This will be investigated in future work.

In contrast to the means and variances, the ML-estimates of the priors have simple closed form solutions. Again there are two choices depending on the form of the representation.

- **ML meta-component priors**: here the auxiliary function $\mathcal{Q}^{(c)}(\mathcal{M}, \hat{\mathcal{M}})$ is optimised subject to the constraints that

[2] For the implementation used in this work the positive variance estimates were found to be highly sensitive to the precise initialisation used.

$$\sum_{\mathbf{m}} c_{\mathbf{m}} = 1; \quad c_{\mathbf{m}} \geq 0. \tag{37}$$

This yields the standard optimisation problem associated with estimating the component priors for a mixture scheme. Thus

$$c_{\mathbf{m}} = \frac{\sum_{t=1}^{T} \gamma_t^{(\mathbf{m})}}{\sum_{t=1}^{T} \sum_{\mathbf{m}} \gamma_t^{(\mathbf{m})}} \tag{38}$$

where the summation in the denominator is over all meta-components of the state.

- **Product meta-component priors**: the individual components must satisfy

$$\sum_{m=1}^{M^{(s)}} c_m^{(s)} = 1; \quad c_m^{(s)} \geq 0 \tag{39}$$

for all streams. The auxiliary function may then be expressed in terms of the stream component priors using equation 17

$$\mathcal{Q}^{(c)}(\mathcal{M}, \hat{\mathcal{M}}) = \sum_{t=1}^{T} \sum_{\mathbf{m}} \gamma_t^{(\mathbf{m})} \sum_{s=1}^{S} \log\left(c_m^{(s)}\right). \tag{40}$$

Again this is closely related to standard ML estimates of the components. The ML estimate of the prior for stream s is

$$c_m^{(s)} = \frac{\sum_{t=1}^{T} \sum_{\{\mathbf{m}:\, m_s = m\}} \gamma_t^{(\mathbf{m})}}{\sum_{t=1}^{T} \sum_{\mathbf{m}} \gamma_t^{(\mathbf{m})}} \tag{41}$$

where the summation in the denominator is over all meta-components of the state.

One issue in training a PoG system is how to appropriately initialise the system. Various approaches are possible (10). For this work, the relationship between the PoG system and the multiple stream system is used, as discussed in the next section. A multiple stream system is built by partitioning the feature vector. The trained multiple stream system is then converted into a PoG system by "padding" the covariance matrix with high cross-stream values [3] and zeroing the cross-stream means. This is similar to the subspace initialisation in (10). This relationship is discussed in more detail in section 5.

[3] In practice a constant times the inverse of the variance floor was used.

5 Multiple Stream Systems

One standard distributed representation, closely related to the scheme examined here, is multiple stream modeling (20). Here, the feature vector is assumed to consist of independently modeled *streams*. Each of the stream subvector observations is generated independently from all the other streams. Observations from these individual streams are then concatenated together to form the full feature vector. For an S-stream system

$$\mathbf{o}_t = \begin{bmatrix} \mathbf{o}_t^{(1)} \\ \vdots \\ \mathbf{o}_t^{(S)} \end{bmatrix} \tag{42}$$

where $\mathbf{o}_t^{(s)}$ is the observation sub-vector associated with stream s. Performance for this form of model is degraded by the independent stream assumption, and to date multiple stream systems have had very limited success when applied to large vocabulary speech recognition tasks.

This section describes these multiple stream systems and relates them to the PoG system described in the previous section. The form of multiple stream system considered here is the synchronous independent stream model implemented in HTK(20). This form of multiple stream model makes the assumption that, given the state, the observations from each of the streams are independent of one another. This may be expressed as

$$p(\mathbf{o}_t|q_t) = \prod_{s=1}^{S} \left(\sum_{m=1}^{M^{(s)}} c_m^{(s)} \mathcal{N}\left(\mathbf{o}_t^{(s)}; \boldsymbol{\mu}_m^{(s)}, \boldsymbol{\Sigma}_m^{(s)}\right) \right). \tag{43}$$

In a similar fashion to the PoG system, this may be expressed in the product space given in equation 20. The meta-component means and variances are now

$$\boldsymbol{\mu}_{\mathbf{m}} = \begin{bmatrix} \boldsymbol{\mu}_{m_1}^{(1)} \\ \vdots \\ \boldsymbol{\mu}_{m_S}^{(S)} \end{bmatrix} \quad \boldsymbol{\Sigma}_{\mathbf{m}} = \begin{bmatrix} \boldsymbol{\Sigma}_{m_1}^{(1)} & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \boldsymbol{\Sigma}_{m_S}^{(S)} \end{bmatrix}. \tag{44}$$

The total number of effective full-dimensional Gaussians is then the number of possible combinations of stream elements, which is the same as for the PoG system. The prior for an effective full-dimensional component is given by the product of the individual stream component priors, shown in equation 17.

The relationship between the PoG system and multiple stream systems is best illustrated by an example. Consider a 2-stream PoG system where the covariance matrices of component one of the two streams are given by

$$\boldsymbol{\Sigma}_1^{(1)} = \begin{bmatrix} \boldsymbol{\Sigma}^{(1)} & \mathbf{0} \\ \mathbf{0} & \sigma^2\mathbf{I} \end{bmatrix}, \quad \boldsymbol{\Sigma}_1^{(2)} = \begin{bmatrix} \sigma^2\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}^{(2)} \end{bmatrix}. \tag{45}$$

For the situation where $\sigma^2 = \infty$, using equation 16 to compute the meta-component variance yields

$$\boldsymbol{\Sigma}_{[11]'} = \begin{bmatrix} \boldsymbol{\Sigma}^{(1)} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}^{(2)} \end{bmatrix}. \tag{46}$$

This may be considered as making some dimensions of the individual experts non-informative. Thus when the "cross-stream" variances for a PoG system are very large, a PoG system becomes the same as a multiple stream system. Similarly, the mean of PoG will have the same form as the multiple stream mean. Figure 3 shows the effect of varying the value of σ^2, the cross-stream variance, on the meta-component positions. The figure shows two MoG experts, for stream 1 the means are at $[2\ 2]'$ and $[\text{-}2\ \text{-}2]'$ and for stream 2 at $[2\ \text{-}2]'$ and $[\text{-}2\ 2]'$. The "within-stream" variances are 1. The top left plot has $\sigma^2 = 1000$ and the resultant meta-components are the same as a multiple stream system. As σ^2 decreases the meta-components are no longer aligned with the axis. For $\sigma^2 = 1$ the components are rotated by almost 45 degrees compared to the $\sigma^2 = 1000$. The PoG system can be seen to be more powerful than the multiple stream system, as there is no assumption about the meta-components aligning with the "axis" of the two streams. However, there is an increase in the number of model parameters, since each expert in a PoG system models the complete feature vector.

In common with a PoG system, it is possible to use ML meta-component priors with a multiple stream system. Multiple stream systems with ML-meta-component priors are related to subspace distributed clustering HMMs (SDCHMMs) (11). In SDCHMMs there is an atom table consisting of S streams and $M^{(s)}$ components for stream s. All components of the system are formed by combinations of elements from the atom table. Each component prior is then calculated in an ML fashion. Using ML components for a multiple stream system is equivalent to an SCDHMM system with distinct atom tables for each state.

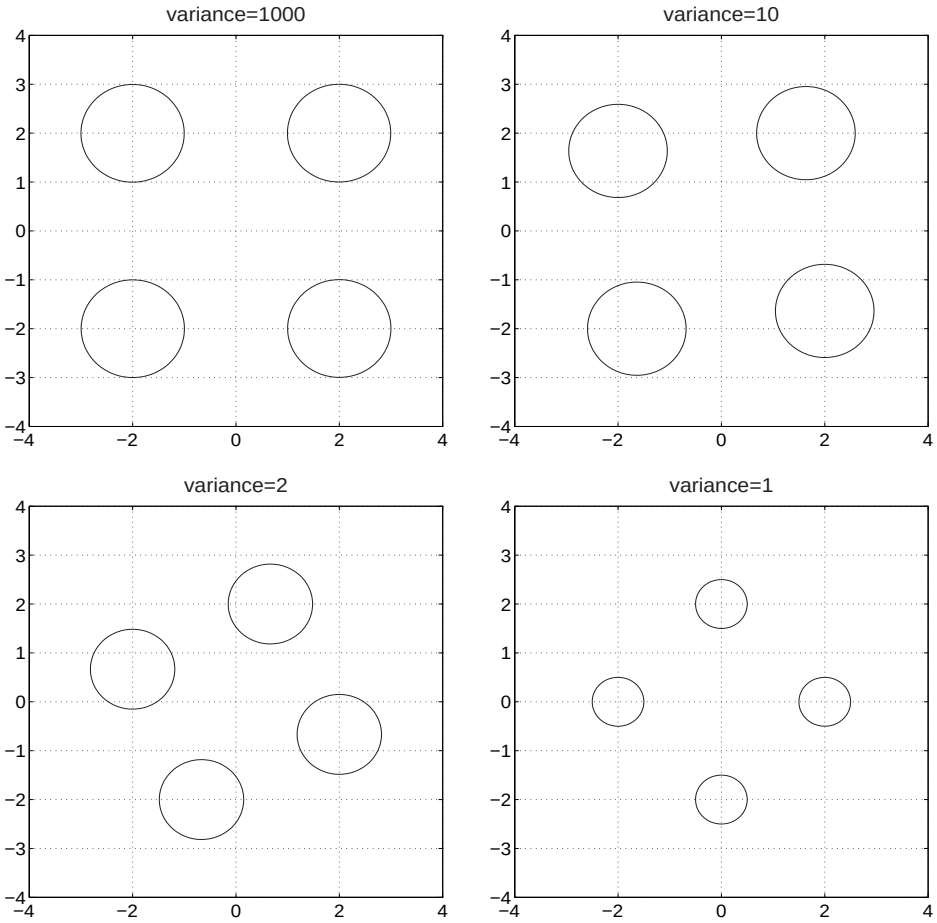

Fig. 3. The circles show one standard deviation contours of each Product of Experts. The products shift as cross-stream variance is reduced.

6 Results

The performance of the PoG and multiple stream systems were evaluated on a standard large-vocabulary speaker-independent speech recognition task. Hub5, or Switchboard. This is a telephone bandwidth spontaneous speech recognition task. The acoustic training data is obtained from two corpora: Switchboard-1 (Swb1) and Call Home English (CHE). The full training corpus consists of an 265-hour training set, 4482 sides from Swb1 and 235 sides from CHE. For the experiments performed in this paper a subset of this was used. A total of 68 hours was chosen to include all the speakers from Swb1 as well as a subset of the available CHE sides. 862 Swb1 sides and 92 CHE sides were used in this subset. This is the `h5train00sub` training set described in (9). The speech waveforms were coded using perceptual linear prediction cepstral coefficients derived from a Mel-scale filterbank (MF-PLP) covering the frequency range from 125Hz to 3.8kHz. A total of 13 coefficients, including c_0, and their first and second order derivatives were used. Cepstral mean subtraction and variance normalisation were performed for each conversation

side. Vocal tract length normalisation (VTLN) was applied in both training and test. A gender-independent cross-word-triphone diagonal-covariance mixture-Gaussian tied-state HMM system was built.

All results are quoted on a three-hour subset of the 2001 development data, referred to as `dev01sub`. This has been found to be a good predictor of system performance. For all recognition experiments single pass decodes were performed, rather than using lattices, to avoid cross system effects. A trigram language model was used built using the language model training data described in (9).

	Number of Components					
System	2	4	6	8	10	12
`std`	46.1	43.7	42.0	40.7	39.3	39.1
`stm`	46.0	43.8	43.1	42.4	42.1	42.0

Table 1
`dev01sub` Switchboard WER using MoG (`std`) and 3-stream multiple-stream (`stm`) systems.

Table 1 shows a direct comparison of a standard MoG HMM system with a 3-stream multiple stream system, the streams were the static, first and second derivative parameters. For small numbers of components there was little performance difference between the two systems. However, as the number of components was increased the performance difference became large. The standard system significantly outperformed the multiple stream system with a 12-component standard system yielding an error rate 2.9% absolute better than the equivalent multiple stream system.

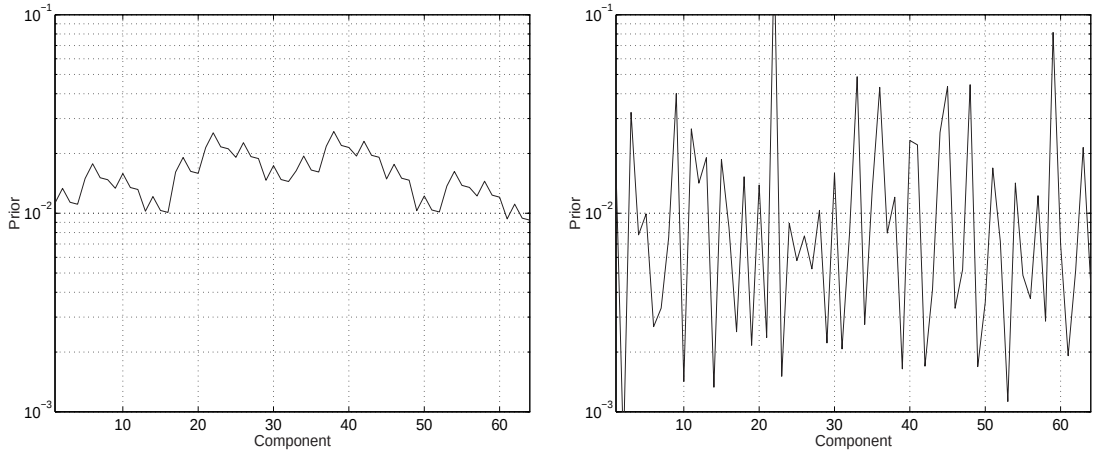

Fig. 4. Product meta-component priors (left) and ML meta-component priors (right) for the 4-component 3-stream system

The difference in the two methods for estimating priors was then evaluated for the multiple stream system. Figure 4 shows, on the left-hand-side, the

product meta-component priors and, on the right-hand-side, the ML meta-component priors for a state of the 4-component multiple stream system. Using ML meta-component priors increases the number of model parameters. In this case the product prior system has nine free parameters per state for the weights while the ML prior system has 63. The ML meta-component priors show much greater variance than the product priors. This indicates that the priors are not well modeled using the standard distributed representation. The performance of the 4-component multiple stream system using the ML meta-component priors was 43.5% error rate compared to 43.8% for the standard multiple stream system. Iteratively training all the model parameters using the ML priors further reduced the error rate to 43.0%. This indicates that a source of degradation of performance of the multiple stream system is the product form of the meta-component priors. Table 2 shows the average training

System	Prior	Var	Log-Lik	Error Rate
`std`	—	—	-69.36	46.1
`stm`	`prd`	—	-69.34	46.0
	`ml`	—	-69.09	45.7
`pog`	`ml`	`+ve`	-68.90	44.4
	`prd`	`-ve`	-68.19	43.2
	`ml`	`-ve`	-68.17	43.1

Table 2
`dev01sub` Switchboard performance using standard (`std`) and 3-stream systems (`stm`) and PoG systems (`pog`), with two components per stream.

data log-likelihood per frame and the recognition performance for a PoG and multiple stream systems using 2-component per stream systems. The priors for the systems are estimated either according to a product of stream experts as given in equation 17, `prd`, or using the ML priors estimated according to equation 38, `ml`. Additionally, the variances for the PoG system are either constrained to be positive, `+ve`, or allowed to be negative, `-ve`. Comparing the product meta-component prior multiple stream system with the standard MoG system, there is little difference between the two log-likelihoods. This may be partly attributed to the limitation of the product form of the components and to the small number of components in the underlying streams. The word error rates of the two systems are approximately the same. Using ML meta-component priors for the multiple stream system yields minor improvements in log-likelihood and word error rate.

The three PoG systems all have higher log-likelihoods and lower word error rates than the `std` and `stm` systems. This is not surprising since the performance of the multiple stream systems indicates that the independent stream assumption is poor for speech recognition with MF-PLP parameters. It also

illustrates that significant use is made of the cross-stream variances to better model the data.

When the variances were constrained to be positive, the log-likelihoods decreased and the error rates increased. Using PoGs with positive variances for each of the experts gave an error rate of 44.1%. When this restriction was relaxed to allow negative variances, the error rate was 43.1%, significantly lower than when the variances were restricted to be positive. Thus the additional flexibility of having "negative" variances is useful for `pog` systems. In contrast, the comparison of the `ml` over `prd` component priors for the PoG system shows rather less impact on performance. However, this may be attributed to the limited number of components per stream.

System	Prior	Var	Log-Lik	Error Rate
`std`	—	—	-68.60	43.7
`stm`	`prd`	—	-68.60	43.8
	`ml`	—	-68.14	43.0
`pog`	`ml`	`-ve`	-66.78	40.9

Table 3
`dev01sub` Switchboard performance using standard (`std`) and 3-stream systems (`stm`) and PoG systems (`pog`), with four components per stream.

To further investigate more complex systems a set of 4-component per stream systems were built. Table 3 shows the performance of these 4-component per stream systems. As the number of components per stream increases, the advantage of using `ml` priors increases (though note the number of model parameters also slightly increases). This is shown in both an increase in log-likelihood and decrease in error rate. Again the use of the `pog` yields lower error rate. However, the additional parameters required by the PoG system must be noted. If the total number of model parameters is considered, rather than the components per stream, the 4-component PoG system, with an error rate of 40.9%, is equivalent to the 12 component MoG system, which had an error rate of 39.1%. The PoG system performance was significantly worse than the MoG performance for approximately the same number of model parameters. However, comparing the average training data log-likelihoods of the two systems, the PoG system is slightly higher, -66.8, compared to the MoG system, -67.1. The 4-component PoG system better models the training data, though it is not a better model for classification.

7 Conclusions

This paper has described a new form of distributed representation, the PoG model, based on the PoE framework. Techniques for training and initialising this new model are presented. In addition, the relationship between this model and a multiple stream model is described. This PoG model is compared to standard MoG and multiple stream state-representations for HMM-based speech recognition. A standard speech recognition task, Switchboard, was used for the evaluation. As expected, the performance of the multiple stream system became worse than that of the MoG system as the number of components increased. Part of this degradation in performance was shown to be due to the poor representation of the priors in the system. Additional experiments on larger systems are required to further evaluate this effect. The performance of the PoG system was better than that of MoG or multiple stream systems with an equivalent number of components per stream. However, the PoG system has more parameters for each component. The largest PoG system trained, 4-components per stream, had a performance significantly worse than that of the best, 12-component, MoG system.

Future work will investigate building larger PoG systems to see whether as the number components increase the performance exceeds the standard MoG system. There are a number of issues that still need to be resolved for the PoG system. As well as efficient training and initialisation schemes, there is the need for efficient decoding schemes and techniques for selecting the number of streams. Future work will also examine the product of MoG HMM systems for speech recognition.

References

[1] S S Airey. Products of Gaussians. Master's thesis, University of Cambridge, 2002.

[2] H Bourlard and S Dupont. A new ASR approach based on independent processing and combination of partial frequency bands. In *Proceedings ICSLP*, pages 422–425, 1996.

[3] A D Brown and Hinton G E. Products of hidden Markov models. Technical Report GCNU TR 2000-08, Gatsby Computational Neuroscience Unit, 2000.

[4] A P Dempster, N M Laird, and D B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

[5] M J F Gales and S S Airey. Product of Gaussians for Speech Recognition. Technical Report CUED/F-INFENG/TR458, Cambridge University, 2003. Available from: svr-www.eng.cam.ac.uk/~mjfg.

[6] M J F Gales. Transformation streams and the HMM error model. *Computer Speech and Language*, 16:225–243, 2002.

[7] M J F Gales and S J Young. Robust speech recognition using parallel model combination. *IEEE Transactions Speech and Audio Processing*, 4:352–359, 1996.

[8] Z Ghahramani and M I Jordan. Factorial hidden Markov models. *Machine Learning*, 29:245–275, 1997.

[9] T Hain, P C Woodland, G Evermann, and D Povey. The CU-HTK March 2000 HUB5E transcription system. In *Proceedings of the Speech Transcription Workshop*, 2000.

[10] G Hinton. Products of experts. In *Proceeding of ICANN*, 1999.

[11] B Mak and E Bocchieri. Subspace distribution clustering for continuous observation density hidden Markov models. In *Proceedings Eurospeech*, pages 107–110, 1997.

[12] H Nock and S Young. (2000). Loosely coupled HMMs for ASR. In *Proceedings ICSLP, 2000.*

[13] P A Olsen and R A Gopinath. Modeling inverse covariance matrices by basis expansion. In Proceedings ICASSP*, 2002.*

[14] A-V I Rosti and M J F Gales. Generalised linear Gaussian models. Technical Report CUED/F-INFENG/TR420, Cambridge University, 2001. Available from: svr-www.eng.cam.ac.uk/~mjfg.

[15] S Roweis and Z Ghahramani. A unifying review of linear Gaussian models. Neural Computation*, 11:305–345, 1999.*

[16] D B Rubin and D T Thayer. EM algorithms for ML factor analysis. Psychometrika*, 47:69–76, 1982.*

[17] L K Saul and M I Jordan. Mixed memory Markov models. Machine Learning*, 37:75–87, 1999.*

[18] M Tipping and C Bishop. Mixtures of probabilistic principal component analyzers. Neural Computation*, 11:435–474, 1999.*

[19] C K I Williams, F V Agakov, and S N Felderhof. Products of Gaussians. In Proceeding of NIPS*, 2001.*

[20] S J Young, J Jansen, J Odell, D Ollason, and P Woodland. The HTK Book (for HTK Version 2.0)*. Cambridge University, 1996.*

[21] G Zweig. Speech Recognition with Dynamic Bayesian Networks*. PhD thesis, ICSI, UC Berkeley, 1999.*