SPEECH MODELING USING VARIATIONAL BAYESIAN MIXTURE OF GAUSSIANS

Panu Somervuo

International Computer Science Institute
Berkeley, California, USA
panus@icsi.berkeley.edu

Neural Networks Research Centre
Helsinki University of Technology, Finland
panu.somervuo@hut.fi

ABSTRACT

The topic of this paper is speech modeling using the Variational Bayesian Mixture of Gaussians algorithm proposed by Hagai Attias (2000). Several mixtures of Gaussians were trained for representing cepstrum vectors computed from the TIMIT database. The VB-MOG algorithm was compared to the standard EM algorithm. VB-MOG was clearly better, its convergence was faster, there was no tendency to overfitting, and finally, it gave consistently better likelihoods for unseen test data using any given number of the mixture components.

1. INTRODUCTION

Mixture densities are commonly used in the feature modeling. The components of the mixtures are typically simple parametric models such as Gaussians. There are various methods for training the models, each with a different optimization criterion. The fundamental problem in the modeling is how to choose the optimal model complexity. In case of the mixture models we may ask what is the optimal number of the mixture components.

Maximum likelihood (ML) estimation is a commonly used training method. However, it suffers from the overfitting if there are too many free parameters in the model compared to the size of the training data. In that case the model fits very well to the training data but lacks the generalization ability so that it doesn't tolerate noise or other deviations from the expected training data. As a consequence, the model cannot be used for making inferences about the new data. Overfitting can be reduced by adding a penalty term to the ML objective function. The basic idea is that the penalty term becomes larger as the model complexity grows. However, some penalty terms are heuristic or valid only for large sample sizes. Another way to control the overfitting is crossvalidation. The original training data is then divided into subsets and the model is trained using all subsets except the one which is saved for the evaluation. This can be repeated with different partitions of the data to the training and validation sets. When comparing the models, that structure is chosen which gives the best performance to the validation data. Although this is straightforward to implement, it can be very time consuming, especially if the number of the partitions is large.

Many estimation methods are based on point densities. For instance, ML estimation gives a single model corresponding to the highest (may be local) peak of the likelihood function. The MAP estimation gives the highest (again possibly local) peak of the posterior probability function. The problem with the point estimates is that they do not measure the probability of the model which in practice is of importance. For instance, the model based on a single density peak can be overfitted. Bayesian modeling gives a more rigorous and disciplined way to carry out the parameter estimation and model selection. In this framework, an entire model class with all its possible parameter combinations are handled together. Sometimes the term Bayesian modeling is used when only adding priors to the ML estimation (i.e. as a synonym to the MAP estimation), however, the main challenge is moving from the models based on point density estimates to the models which are averaged over the entire parameter posterior probability distribution. This is the way the Bayesian modeling is applied in the current work.

In the experiments, cepstrum vectors computed from speech were modeled using the mixtures of Gaussians. Bayesian framework was used for training and comparing the models. When using the models in the full Bayesian setting, all models with different structures can be used together for making inferences about new data. The value of the well-defined objective function can also be used for selecting one best model structure, in this case the optimal number of the mixture components.

2. VB-MOG ALGORITHM

In the paper of Hagai Attias [1], by means of the calculus of variations, an expectation maximization (EM) type learning algorithm is derived for probabilistic graphical models. In particular, an algorithm for training a mixture of Gaussians was presented. This algorithm, Variational Bayesian Mixture of Gaussians (VB-MOG) is described below.

Let $Y = \{\mathbf{y}_1, \ldots \mathbf{y}_N\}$ denote the N visible data observation nodes, $X = \{\mathbf{x}_1, \ldots \mathbf{x}_N\}$ the hidden nodes, and Θ the parameters (additional hidden nodes) of a Bayesian network. A structure parameter m controls the number of the hidden nodes (the number of the mixture components). The joint distribution $p(Y, X|\Theta, m)$ defines now a model.

The Gaussian mixture model with m mixture components is expressed as:

$$p(\mathbf{y}_n|\Theta, m) = \sum_{s=1}^{m} p(\mathbf{y}_n|s_n = s, \Theta) p(s_n = s|\Theta) \,, \qquad (1)$$

where $\mathbf{y}_n$ denotes the nth data vector and s_n the hidden component that generated it. Each mixture component is a Gaussian, $p(\mathbf{y}_n|s_n = s, \Theta) = \mathcal{N}(\boldsymbol{\mu}_s, \boldsymbol{\Gamma}_s)$, where $\boldsymbol{\mu}_s$ is the mean, $\boldsymbol{\Gamma}_s$ is the precision matrix (inverse of the covariance matrix $\boldsymbol{\Sigma}_s$), and $p(s_n = s|\Theta) = \pi_s$ is the mixture weight.

This work was supported by the Academy of Finland, project no. 44886 "New information processing principles" (Finnish Centre of Excellence Programme 2000-2005).

In Bayesian modeling we try to compute the parameter posterior distribution $p(\Theta|Y,m)$. Since this is in general intractable, the true posterior is approximated by a tractable form q. In the approximation, the parameters and hidden nodes are assumed to be independent so that q can be factorized: $q(X,\Theta|Y) = q(X|Y)q(|\Theta|Y)$.

In the Variational Bayesian framework, the model parameters need not be assumed to be e.g. normally distributed, instead, the functional form of q results from the free-form optimization of an objective function. This optimization gives the best approximation to the true posterior within the space of distributions having the factorized form. If the parameter priors have the same functional forms as the parameter posteriors in q, they are called conjugate priors. For parameters $\Theta = \{\pi_s, \boldsymbol{\mu}_s, \boldsymbol{\Gamma}_s\}$ the following priors are used: the mixture weights are jointly Dirichlet, $p(\{\pi_s\}) = \mathcal{D}(\lambda^0)$, the means conditioned on the precisions are Normal, $p(\boldsymbol{\mu}_s|\boldsymbol{\Gamma}_s) = \mathcal{N}(\boldsymbol{\rho}^0, \beta^0\boldsymbol{\Gamma}_s)$, and the precisions are Wishart, $p(\boldsymbol{\Gamma}_s) = \mathcal{W}(\nu^0, \boldsymbol{\Phi}^0)$. In [1] Wishart distributions were used, but in the present work where the data vectors consist of cepstral coefficients, diagonal covariance matrices can be used for the Gaussian mixture components (cepstral coefficients are fairly decorrelated). In this case the Wishart distribution can be replaced by the product of the Gamma distributions $\prod_{i=1}^{d} \mathcal{G}(a_i, b_i)$, where $a_i = \nu^0/2$, $b_i = \Phi_i^0/2$, Φ_i^0 denoting the ith diagonal element of the matrix $\boldsymbol{\Phi}^0$, and d is the dimension of the data vector (notice that the Gamma distribution is sometimes defined as $\mathcal{G}(a, 1/b)$ instead of $\mathcal{G}(a,b)$).

The objective function to be maximized is:

$$\mathcal{F}_m = \int q(X)q(\Theta) \ln \frac{p(Y,X|\Theta)}{q(X)q(\Theta)} dX d\Theta \,, \tag{2}$$

where q is conditioned on Y (although not explicitly shown). The EM-like algorithm [1] consists of two steps: In the E-step, the posterior over the hidden nodes is computed by solving $\delta\mathcal{F}_m/\delta q(X) = 0$ to get:

$$q(X) \propto e^{\int q(\Theta) \ln p(Y,X|\Theta) d\Theta} \,. \tag{3}$$

In case of the mixture model (1) where the Gaussians have diagonal covariances, the responsibility of the mixture component s for the data vector $\mathbf{y}_n$ is:

$$\begin{aligned} \gamma_s^n \quad = \quad & q(s_n = s|\mathbf{y}_n) \propto \exp\Bigg(\Psi(\pi_s) - \Psi(\sum_{s'} \pi_{s'}) \\ & + \frac{d}{2}\left(\Psi(\frac{\nu_s}{2}) - \ln 2\pi - \beta_s^{-1}\right) \\ & - \frac{1}{2}\sum_{i=1}^{d} \ln\frac{\Phi_{si}}{2} + \nu_s \Phi_{si}^{-1}(\mathbf{y}_{ni} - \boldsymbol{\rho}_{si})^2 \Bigg) \,, \end{aligned} \tag{4}$$

where $\mathbf{y}_{ni}$ and $\boldsymbol{\rho}_{si}$ denote the ith component of the vectors, $\boldsymbol{\Phi}_{si}$ is the ith diagonal component of matrix $\boldsymbol{\Phi}_s$, and Ψ is the digamma function $d\ln\Gamma(z)/dz$. The responsibilities sum to unity: $\sum_s \gamma_s^n = 1$ for all n.

In the M-step the posterior distribution over the parameters is computed by solving $\delta\mathcal{F}_m/\delta q(\Theta) = 0$ to get:

$$q(\Theta) \propto e^{\int q(X) \ln p(Y,X|\Theta) dX} p(\Theta) \,. \tag{5}$$

The parameter posterior is computed in two stages: First the parameters Θ are updated:

$$\begin{aligned} \bar{\pi}_s &= \frac{1}{N}\sum_{n=1}^{N} \gamma_s^n \,, \qquad \bar{\boldsymbol{\mu}}_s = \frac{1}{\bar{N}_s}\sum_{n=1}^{N} \gamma_s^n \mathbf{y}_n \,, \\ \bar{\boldsymbol{\Sigma}}_s &= \frac{1}{\bar{N}_s}\sum_{n=1}^{N} \gamma_s^n (\mathbf{y}_n - \bar{\boldsymbol{\mu}}_s)(\mathbf{y}_n - \bar{\boldsymbol{\mu}}_s)^T \,, \end{aligned} \tag{6}$$

where $\bar{N}_s = N\bar{\pi}_s$. Then the posterior parameters are updated:

$$\begin{aligned} \lambda_s &= \bar{N}_s + \lambda^0 \,, \qquad \boldsymbol{\rho}_s = (\bar{N}_s\bar{\boldsymbol{\mu}}_s + \beta^0\boldsymbol{\rho}^0)/(\bar{N}_s + \beta^0) \,, \\ \beta_s &= \bar{N}_s + \beta^0 \,, \qquad \nu_s = \bar{N}_s + \nu^0 \,, \\ \boldsymbol{\Phi}_s &= \bar{N}_s\bar{\boldsymbol{\Sigma}}_s + \bar{N}_s\beta^0(\bar{\boldsymbol{\mu}}_n - \boldsymbol{\rho}^0)(\bar{\boldsymbol{\mu}}_n - \boldsymbol{\rho}^0)^T/(\bar{N}_s + \beta^0) + \boldsymbol{\Phi}^0 \,. \end{aligned} \tag{7}$$

The training proceeds by iterating the equations (4), (6), and (7). The final values of the posterior parameters are the result of the algorithm and they can be used for making inferences about new data $\mathbf{y}$. Integrating out the parameters of the model results in the predictive density [1]:

$$p(\mathbf{y}|Y) = \sum_{s=1}^{m} \bar{\pi}_s t_{\omega_s}(\mathbf{y}|\boldsymbol{\rho}_s, \Lambda_s) \,, \tag{8}$$

where each mixture component is a Student-t distribution with the degrees of freedom $\omega_s = \nu_s + 1 - d$, the mean $\boldsymbol{\rho}_s$, the covariance $\Lambda_s = ((\beta_s+1)/\beta_s\omega_s)\boldsymbol{\Phi}_s$, and mixture weight $\bar{\pi}_s = \lambda_s/\sum_{s'}\lambda_{s'}$.

The objective function which is needed for comparing the models is derived next. In the paper of Attias (2000) this was omitted. However, without the value of the objective function, the model comparison is not possible. The objective function (2) can be written as

$$\begin{aligned} \mathcal{F}_m &= \int q(X)q(\Theta) \ln \frac{p(Y,X|\Theta)}{q(X)} dX d\Theta - \int q(\Theta) \ln \frac{q(\Theta)}{p(\Theta)} d\Theta \\ &= \int q(X)q(\Theta) \ln \prod_{n=1}^{N} p(\mathbf{y}_n, \mathbf{x}_n|\Theta) dX d\Theta \\ &\quad - \int q(X) \ln q(X) dX - \int q(\Theta) \ln q(\Theta) d\Theta \\ &\quad + \int q(\Theta) \ln p(\Theta) d\Theta \,. \end{aligned} \tag{9}$$

The first term in the righthand side of (9) equals to

$$\sum_{n=1}^{N}\sum_{s=1}^{m} \int q(\Theta) \ln\left(p(\mathbf{y}_n|s_n = s, \Theta) p(s_n = s|\Theta)\right) d\Theta \,, \tag{10}$$

where the integral is the same that occurs in (3) and has been evaluated in (4) (but now no exponentials are used and the sums of the terms over s are not normalized to unity). The second term is

$$-\int q(X) \ln q(X) dX = -\sum_{n=1}^{N}\sum_{s=1}^{m} q(s_n = s) \ln q(s_n = s) \,, \tag{11}$$

where $q(s_n = s) = \gamma_s^n$ from (4) (note that γ_s^n sum to unity). For the two remaining terms we need to evaluate the integrals having the form $\int f_A \ln f_B$, where f_A and f_B are Gaussians $\mathcal{N}$, Gamma

distributions $\mathcal{G}$, and Dirichlet distributions $\mathcal{D}$. The following three expressions are needed for these cases:

$$\int \mathcal{N}(z;\mu_A,\sigma_A)\ln\mathcal{N}(z;\mu_B,\sigma_B)dz = -\frac{1}{2}\ln 2\pi\sigma_B^2 - \frac{(\mu_A-\mu_B)^2+\sigma_A^2}{2\sigma_B^2}\,, \quad (12)$$

$$\int \mathcal{G}(z;a_A,b_A)\ln\mathcal{G}(z;a_B,b_B)dz = a_B\ln b_B - \ln\Gamma(a_B) + (a_B-1)(\Psi(a_A)-\ln b_A) - b_B a_A/b_A\,, \quad (13)$$

and

$$\int \mathcal{D}(z;\{\alpha_{As}\})\ln\mathcal{D}(z;\{\alpha_{Bs}\})dz = -\sum_{s=1}^{m}\ln\Gamma(\alpha_{Bs}) + \ln\Gamma(\sum_{s=1}^{m}\alpha_{Bs}) + \sum_{s=1}^{m}(\alpha_{Bs}-1)\left(\Psi(\alpha_{As}) - \Psi(\sum_{i=1}^{m}\alpha_{Ai})\right). \quad (14)$$

For d-dimensional vectors, (12) and (13) are computed componentwise and the partial results are summed (since in this work diagonal Gaussians were used). These expressions are computed using priors $p(\Theta)$ and posteriors $q(\Theta)$ as indicated in the last two terms of (9) and summed over all mixture components (notice the factorization $\ln q(\Theta) = \ln q(\{\pi_s\}) + \sum_s \ln q(\boldsymbol{\mu}_s|\boldsymbol{\Gamma}_s) + \ln q(\boldsymbol{\Gamma}_s)$).

2.1. Implementational issues

Equations (4), (13), and (14) require the computation of the digamma function Ψ. This can be implemented using the algorithm [2].

Equations (13) and (14) contain the logarithm of the gamma function. The gamma function and the logarithm should not be computed separately; the result of the gamma function may lead to overflow. Instead, a combined function should be used, e.g. `gammaln` in `MATLAB` or `lgamma` when using the C math library.

Equation (4) may cause numerical problems due to the terms $\nu_s\Phi_{si}^{-1}(\mathbf{y}_{ni}-\boldsymbol{\rho}_{si})^2$ which relate to the weighted distance between the kernel mean and the data vector. In the beginning of the training, $\boldsymbol{\Phi}_s$ is equal to the prior $\boldsymbol{\Phi}^0$. If flat priors are used, Φ_{si} have small values (corresponding to large variances in the Gaussians) and the weighted distances can be large. This may result γ_s^n to be zero due to the underflow (the exponential is taken from a large negative number). If this occurs to γ_s^n with all values of s, the sum of γ_s^n cannot be normalized to unity (division by zero occurs). A remedy for this is to use a proper prior (not too small).

3. EXPERIMENTS

Conventional 12-dimensional cepstrum vectors were extracted from the middle parts of the labeled phones of the TIMIT training database, one feature vector from each phone segment. The resulting data set was then divided into three sets: 10,000 and 100,000 data vectors in the first two sets, and the remaining 61,271 data vectors in the third set. Data vectors were then separately scaled to have zero-means and unit-variances in each set.

Two sets of experiments were conducted, one when using the 10,000 training data vectors and another when using 100,000 vectors for training the Gaussian mixture model. The following priors were used: λ^0, β^0, and ν^0 were equal to one, $\boldsymbol{\rho}^0$ was a 12-dimensional vector with all components zero, and $\boldsymbol{\Phi}^0$ was a unity matrix. Each mean vector of the Gaussian mixture component was initialized by the k-means algorithm [3]. For the smaller training data set, the VB-MOG algorithm was then iterated until the relative change of the objective function was smaller than 10^{-5}. The EM algorithm needed considerably more iterations for the convergence compared to the VB-MOG algorithm. For the larger training set the maximum number of iterations was limited to 20 because of the computational resources. For each model, twenty different initializations were used. Fig. 1 shows the values of the objective function for models with different number of mixture components.

For comparison to the VB-MOG, also a standard ML-based EM-algorithm [4] was applied. The same initializations were used for the EM algorithm as for the VB-MOG. In the EM-algorithm, the mixture weights were initialized equally to the values $1/m$. For variances, a small floor value for used for preventing them to go near zero. The loglikelihood of the training data was penalized by the Bayesian Information Criterion (BIC), $-0.5k\ln N$, where k is the number of the free parameters in the model and N is the number of the training vectors [5], see Fig. 2.

Finally, it was tested how the models perform for unseen data. Average loglikelihoods of the third data set with 61,271 vectors using models trained by VB-MOG and EM algorithm are shown in Fig. 3. It can be seen that both methods underestimate the optimal model complexity but the results obtained by the VB-MOG are consistently better.

4. CONCLUSIONS

Gaussian mixture models are commonly used in the speech modeling. In the current work an algorithm based on the Variational Bayesian framework, the VB-MOG [1], was applied to the training of the models. For comparison, a conventional ML-based EM-algorithm was also used. For optimizing the number of the mixture components in the ML-training, BIC penalization was used. This is, however, valid only for large sample sizes. VB-MOG has a better-defined objective function. In the experiments with the cepstrum data, both methods seemed to underestimate the number of the mixture components which gave the best performance for unseen data. The VB-MOG algorithm performed much better, though. In addition, it converged faster and gave better likelihoods for the test data using any given number of the mixture components.

5. REFERENCES

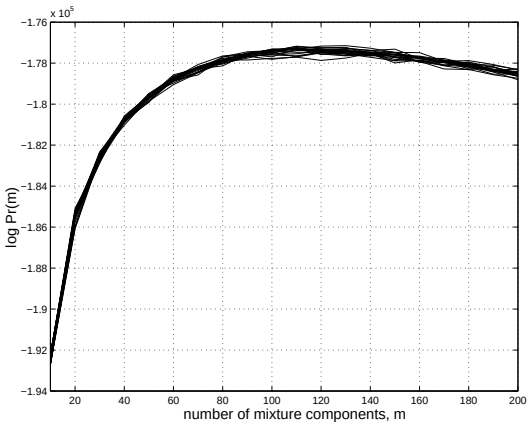
[1] Hagai Attias, "A variational bayesian framework for graphical models," in *Advances in Neural Information Processing Systems (NIPS)*, T. Leen et al, Ed., Cambridge, 2000, vol. 12, pp. 49–52, MIT Press.

[2] J. Bernardo, "As103 psi (digamma) function," *Journal of the Royal Statistical Society (series C) Applied Statistics*, vol. 25, no. 3, pp. 351–317, 1976.

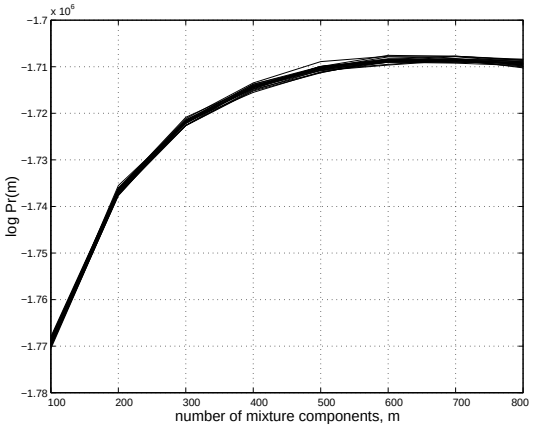
[3] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, vol. COM-28, pp. 84–95, 1980.

[4] A. Dempster, N. Lard, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, B*, vol. 39, no. 1, pp. 1–38, 1977.

[5] G. Schwartz, "Estimating the dimension of a model," *Annals of Statistics*, vol. 6, pp. 461–464, 1978.

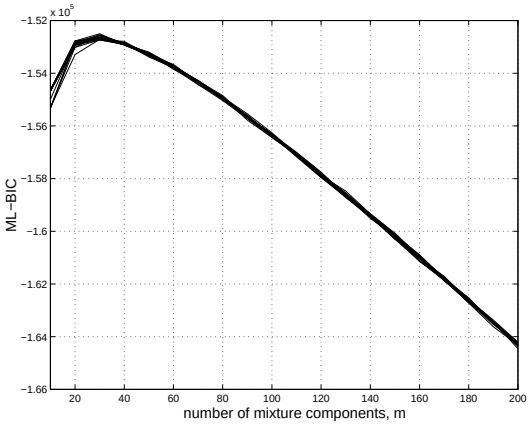


(a) 10,000 training vectors

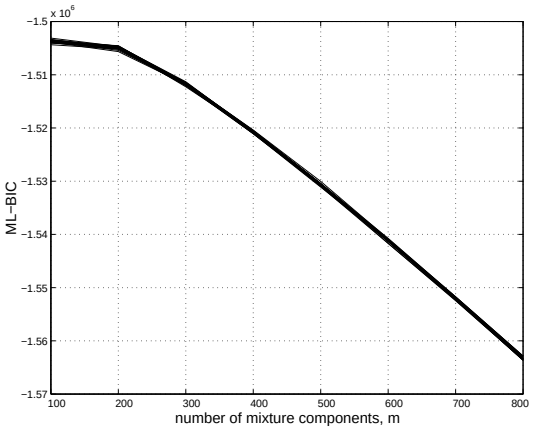


(b) 100,000 training vectors

Fig. 1. Objective function (9) of VB-MOG. Twenty plots correspond to the results from twenty different initializations. The largest mean value of the objective function is around 110 mixture components for a 10,000-vector training set and 700 for a 100,000-vector training set.

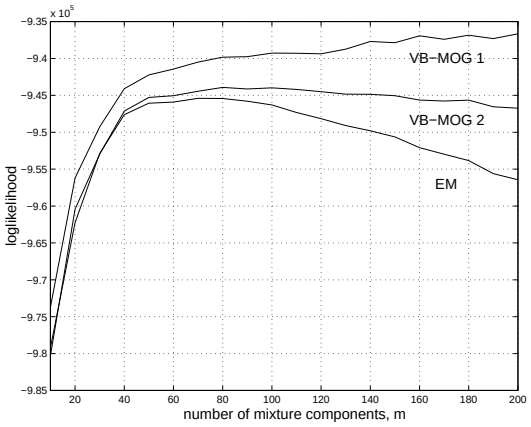


(a) 10,000 training vectors

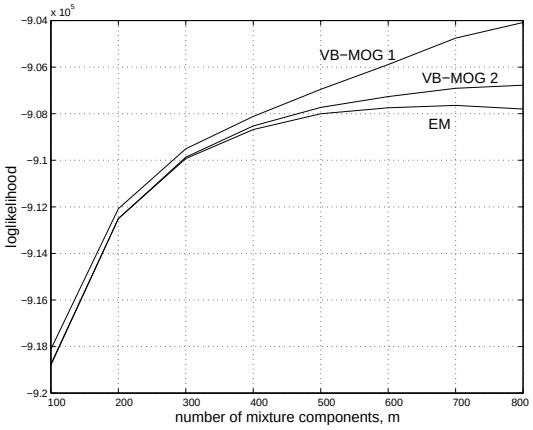


(b) 100,000 training vectors

Fig. 2. BIC-penalized loglikelihood for EM models. Twenty different initializations were used. ML-BIC suggests to use only 30 mixture components for the smaller training data set and 100 (or less) for the larger set.



(a) 10,000 training vectors



(b) 100,000 training vectors

Fig. 3. Average loglikelihood of test data using VB-MOG and EM trained models. VB-MOG 1 corresponds to the predictive density (8) with Student-t distributions and VB-MOG 2 corresponds to the mixture model with Gaussian components (i.e. without model averaging). The results of VB-MOG 1 are consistently better using any number of the mixture components.