ONLINE TRAINING METHODS FOR GAUSSIAN MIXTURE VECTOR QUANTIZERS

Ethan R. Duni and Bhaskar D. Rao

Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, CA 92093-0407
Email: ethan.duni@gmail.com, brao@ucsd.edu

ABSTRACT

This paper presents techniques relevant to the online training of Gaussian Mixture Vector Quantizer (GMVQ) systems. Techniques for learning from quantized data are considered, which enables online training configurations wherein the training is carried out remotely from the encoder. Next, methods for recursive training are presented, which eliminate the need to store large databases of example data, and also enable adaptive operation of the GMVQ system. These techniques are demonstrated on the problem of wideband speech spectrum quantization, and the performance losses due to the use of quantized training data are experimentally quantified as a function of the bit rate.

Index Terms— Quantization, Speech coding, Speech communication, Adaptive systems, Recursive estimation

1. INTRODUCTION

This paper presents techniques for use in the context of online training of Gaussian Mixture Vector Quantizer (GMVQ) systems. To demonstrate the potential of such an approach, we consider it in the context of speaker-dependent wideband speech spectrum coding. Speaker-dependent systems have been used in a variety of speech processing applications. Conventional approaches to speech coding operate in a speaker-independent manner, employing a single coder designed to work for any speaker. This conventional approach has the advantage of simplicity: only one coder needs to be designed, and the design can be carried out ahead of time using a single large, multispeaker database. However, since the statistics of various coder parameters vary widely from speaker to speaker, speaker-dependent coding offers the promise of improved performance. The development of the GMVQ system (see [1] and Fig. 2) provides a flexible coding framework that is able to incorporate arbitrary source statistics and distortion measures. This framework, then, enables the study and implementation of speaker-dependent coding. In our recent work (see [2]), we quantified the gains available in various common speech coder parameters; in particular, we found a gain of 4 bits per frame can be achieved for the Line Spectral Frequencies (LSF). However, since it is impractical to collect an example database of every possible speaker ahead of time, the training process for a real speaker-dependent system must instead be implemented in an online fashion. One on-line training configuration of particular interest is depicted in Figure 1. In this approach, both the encoder and decoder perform the training process in a synchronized way, eliminating the need to explicitly transmit speaker-dependent designs. However, in order for this approach to work, it is necessary to perform the training on data that has been quantized (presumably by a speaker-independent coder, or some other sub-optimal system). Also, it is desirable for the training buffer size to be as small as possible. This paper examines both of these issues in detail. First, however, a bit of background on GMVQ is in order.

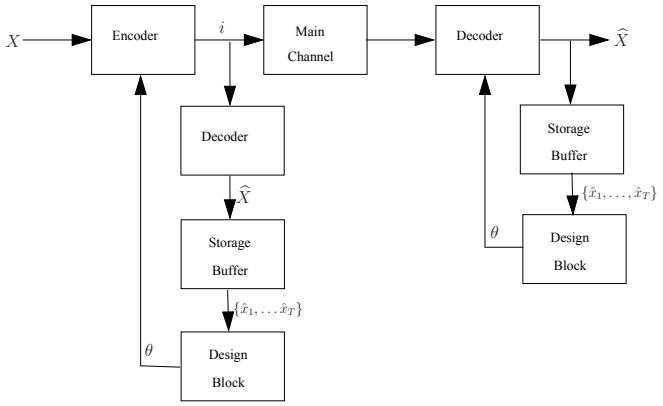

Fig. 1. Synchronized Learning. This configuration avoids the use of side information.

In order to realize speaker-dependent coding, we require a sufficiently general class of quantizers to represent the statistical variations between speakers. To this end, we utilize the GMVQ system presented in [1] (see Figure 2). In addition to the ability to represent reasonably arbitrary source statistics, this system has a number of properties that make it attractive in the speaker-dependent context. Chief among these is the parametric form of the coder design: only a small number of rate-independent parameters are required to describe the coder. For a GMVQ of order M, operating in dimension d, this consists of $M(1 + d + d(d + 1)/2) - 1$ scalar parameters. Thus, the number of parameters that must be transmitted and stored for each speaker is small, and independent on the rate of operation. Note that the values of M typically used in coding are in the range of 10-20, making it feasible to train entire speaker-dependent systems.

Standard training methods for the GMVQ system rely on the EM algorithm. That is, a large database of example data is collected, and then the EM algorithm is applied to provide a set of Gaussian Mixture parameters, which can then be translated into parameters for the GMVQ system (see [1]). However, this approach poses a number of problems in the context of speaker-dependent coding. First, it may be required to perform the training at a location remote from the

This research was supported by Micro Grants 05-033 and 06-174, sponsored by QUALCOMM Inc.

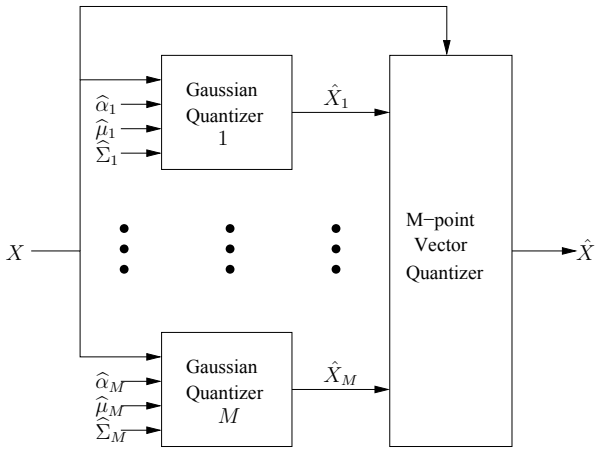

Fig. 2. The Gaussian Mixture Vector Quantizer system. Each component quantizer produces a Gaussian point density $N(x|\mu_m, \Sigma_m)$. The parameters α_m specify the proportion of codepoints allocated to each component quantizer, resulting in an overall point density of $\sum_{m=1}^{M} \alpha_m N(x|\mu_m, \Sigma_m)$.

end-user's equipment, as in Figure 1. Such a training scheme would necessarily have to rely on data that has already been quantized. Problems associated with quantized training data in the context of GMVQ are explored in the Section II, and a post-processing technique is presented to alleviate them. Then, the performance degradation due to quantized training data is experimentally quantified as a function of encoding rate. Next, the requirement of using a large database may result in unacceptable storage requirements. To this end, Section III considers a recursive learning technique, using the Online EM algorithm ([3],[4]) and a learning schedule devised by Sato ([5]). Recursive learning greatly reduces the storage requirements associated with the training process, with negligible impact on performance. Section IV contains a discussion of the results.

2. LEARNING FROM QUANTIZED DATA

A number of issues arise when considering learning from quantized data. First, since the learning process does not have access to clean data, some performance degradation in the resulting design is expected. This loss, as a function of encoding rate, is quantified later in this section. Another issue that arises in the context of GMVQ results from the nature of the quantization error (as opposed to its magnitude as such). To see this, recall that the GMVQ typically utilizes scalar transform coders to implement the component Gaussian coders. The covariance matrices of the individual Gaussians tend, in practice, to be fairly oblong. This results in transform codebooks that, at standard operating rates, lie in subspaces. That is, only a single codepoint is allocated to the least significant transform dimension(s). Note that the entire GMVQ codebook does not lie in a subspace, as the subspaces of each component Gaussian coder do not typically coincide. Nevertheless, attempting to learn a GMVQ of comparable order as was used to quantize the data results in each component "locking on" to a corresponding subspace. This leads to a numerical instability wherein the covariances shrink without bound, derailing the learning process.

To circumvent this problem, a postprocessing can be applied to the quantized data before it is utilized in the learning process. This postprocessing consists of adding Gaussian noise to the quantized data in order to ensure that the data has full rank. Specifically, noise is added only to those coefficients of the (transformed) quantized vector which lie in a subspace (i.e., the dimensions which received an allocation of 1 codepoint). The encoder-decoder pair for a transform coder with postprocessing is illustrated in Figure 3. The variance of the noise for each coefficient is set according to the GMVQ used to quantize it. Note that this postprocessing should not be applied to the actual output of the quantizer in the operational speech coder, as it amounts to adding extra noise: it is only intended to be applied to data for use in the learning process.

In order to demonstrate the effectiveness of the postprocessing scheme, and to quantify the loss due to learning on quantized data, a set of experiments on LSF quantization were performed. These experiments were similar to those used in [2], relying on a database of 45 speakers. For each speaker, a training set of 16-th order wideband LSF parameters was computed, as well as an independent test set. A subset of each speaker's training set was then used to construct a speaker-independent training set. In this experiment, the first step was to train a speaker-independent GMVQ using the clean speaker-independent training set via the usual EM algorithm, with $M = 16$. Then, the training set for each speaker was quantized using the resulting speaker-independent GMVQ in order to provide a quantized training set. Postprocessing as shown in Figure 3 was then employed to ensure that the quantized data was full rank (every cluster had at least one dimension with an allocation of 0 bits). Then, a speaker-dependent GMVQ was trained for each speaker by using the EM algorithm on the quantized training sets, again with $M = 16$. In order to characterize the effects of the encoding rate upon the learning process, the training process was carried out repeatedly using a wide variety of quantization rates. The results are illustrated in Figure 4. Note that the operating point for transparent quality (around 40 bits) incurs about 1 bit per frame of penalty, and lies in a steep section of the performance curve, which is to say that large gains in the performance of speaker-dependent systems can be obtained by increasing the bit rate during training. As the training rate approaches 70-80 bits per frame, the slope levels off. This reflects the fact that such rates are sufficiently large as to make postprocessing unnecessary (i.e., all components of all clusters are allocated at least 1 bit). On the other hand, at very low rates, only a small improvement is possible. In this regime, many components of every cluster receive allocations of 0 bits, and so postprocessing is applied to a large proportion of the components. Since the postprocessing is based on the speaker-independent model, it imposes, to some degree, the speaker-independent statistics onto the training data, resulting in performance very close to the speaker-independent case. Moreover, this curve suggests that it may be beneficial to boost the quantization rate during training, in order to avoid performance penalties. If the rate at which LSFs are coded can be doubled for a time, models trained on quantized data will show very little loss. If desired, this temporary increase in rate could be accomplished without raising the overall rate of the coder by rededicating bits from, say, the fixed codebook during the LSF learning phase. While this would result in degraded audio quality during the training phase, it would also result in a much better speaker-dependent model once training was completed (at which time, the normal bit allocation could be restored).

3. RECURSIVE LEARNING

Usual training strategies for GMVQ assume that the training process has access to a suitably large database of training data, to which the standard EM algorithm is then applied. However, it may be that the requirement of storing a speaker-dependent database is prohibitive in

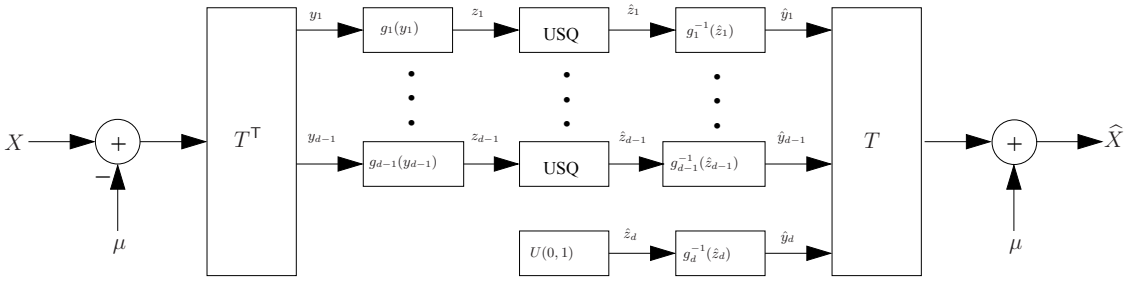

Fig. 3. Transform coder with decoder modified for use in learning. In this example, the d-th transform component received an allocation of 0 bits; all other components are coded as normal. The functions g_i are the compressor functions (i.e., cdf's of Gaussians), the "USQ" blocks are uniform scalar quantizers (on $[0, 1]$) and the $U(0, 1)$ block is a random number generator, uniformly distributed on $[0, 1]$.

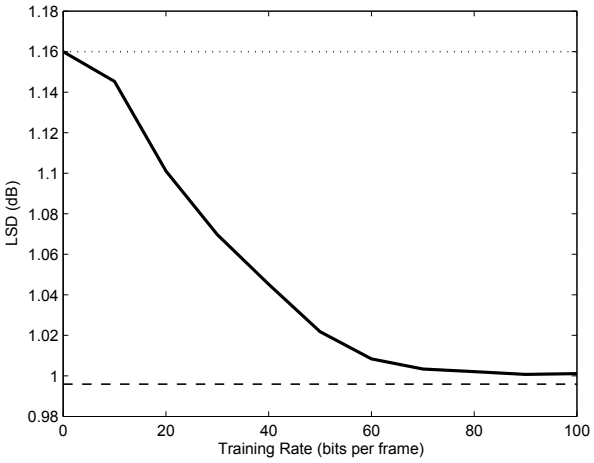

Fig. 4. Performance of Speaker-Dependent LSF Quantizers Trained on Quantized Data, as a function of bit rate. The dashed line shows the performance of speaker-dependent systems trained on clean data, while the dotted line shows speaker-independent performance. All of the systems illustrated operated at a rate of 38 bits per frame.

the context of online learning (for example, if the learning process is to be performed by the end-user's equipment). In these cases, recursive learning can be employed, wherein the learning process utilizes only a single frame at a time, resulting in a sequence of parameter estimates. Such a scheme has very low storage requirements, needing only the current frame's data, the parameters, and a few auxiliary variables used in the recursion. It can also provide for adaptive operation. In such a case, the training process would continue indefinitely, allowing the coders to track changes in the speaker or acoustic environment. In the context of learning a GMVQ, the Recursive EM algorithm can be employed for this purpose. This algorithm, presented by Titterington in [3] and [4], is based on Stochastic Approximation. That is to say, the parameter estimate at time step n takes the form:

$$\theta(n) = \theta(n-1) + \eta(n) T^{-1}(n) \left[\frac{\partial}{\partial \theta} \log f_\theta(x_n) \right]_{\theta=\theta(n-1)} \quad (1)$$

where $\eta(n)$ is a step-size parameter and $T(n)$ is a conditioning matrix. Given certain technical conditions, stochastic approximation theory guarantees that such an estimator is consistent. In particular, it is required that $\sum_n \eta(n) = \infty$ and $\sum_n \eta^2(n) = 0$, i.e., $\eta(n) = o(\frac{1}{n})$. Beyond its effects on consistency, the choice of $T(n)$ determines the relative asymptotic efficiency of the estimation procedure, with optimal performance achieved by employing the Fischer Information matrix (i.e., the Hessian of the log-likelihood). However, in nontrivial problems such as estimation of the parameters of a multivariate GMM, it is very expensive to compute and, particularly, invert the Fischer Information. A popular alternative, then, is to use the "complete-data" Fischer Information matrix, which is much simpler to compute and invert, although it results in decreased relative efficiency. This is referred to as the Recursive EM Algorithm, and results in the following update procedure for the case of GMM:

$$\begin{aligned}
r_{mn} &\propto \alpha_m(n-1) N(x_n | \mu_m(n-1), \Sigma_m(n-1)) && (2)\\
\alpha_m(n) &= (1-\eta(n))\alpha_m(n-1) + \eta(n) r_m n && (3)\\
&= \tau_{mn} + \rho_{mn} && (4)\\
\mu_m(n) &= \frac{\tau_{mn}\mu_m(n-1) + \rho_{mn} x_n}{\tau_{mn} + \rho_{mn}} && (5)\\
\Sigma_m(n) &= \frac{\tau_{mn}\Sigma_m(n-1) + \frac{\tau_{mn}\rho_{mn}}{\tau_{mn}+\rho_{mn}} \langle x_n - \mu_m(n-1) \rangle}{\tau_{mn} + \rho_{mn}} && (6)
\end{aligned}$$

where $\langle . \rangle$ denotes the outer product and τ_{mn} can be thought of as the "prior strength" assigned to the m-th old estimate (as of time $n-1$) and ρ_{mn} represents the new information for cluster m at time n. Notice the similarity between these recursions and the expressions that arise in Sequential MAP estimation of a GMM with a complete-data conjugate prior (see [6]). In the case that $\eta(n) = \frac{1}{n}$, then, the two approaches are equivalent. However, as will be seen shortly, other choices of $\eta(n)$ are more appropriate to the recursive learning problem, in which case the equivalence with MAP estimation does not apply. Also note that the inverse and determinant of $\Sigma_m(n)$ will also be required in order to compute r_{mn} at the each time step. Because the update to Σ_m in Eq. (6) is a rank-one update, these quantities can be efficiently computed in a recursive manner by applying the Matrix Inversion Lemma (see [7] for details).

The last issue to determine is the schedule of the stepsize $\eta(n)$. The simplest approach is simply to utilize $\eta(n) = \frac{1}{n}$. This approach is useful in, for example, adaptive settings wherein a good estimate is already available, and it is desired to update it using new data

(c.f. [8]). In such a scenario, the "prior weight" given to the initial estimate is quite high, and a small stepsize is desired. Thus, a $\frac{1}{n}$ schedule, starting at some fairly large n_0 is appropriate. However, in "from scratch" estimation problems, the simple $\frac{1}{n}$ schedule suffers from its sensitivity to early data. That is, during the early stages of estimation, when the old parameter estimate is very inaccurate, employing a $\frac{1}{n}$ schedule typically causes the estimation procedure to diverge. To avoid this problem, one can employ a modified learning schedule as suggested by Sato in [5]:

$$\eta(n) = \left(\sum_{t=1}^{n} \prod_{s=t+1}^{n} \lambda(s) \right)^{-1} \quad (7)$$

$$\lambda(n) = 1 - \frac{1}{(n-2)\gamma + \frac{1}{\epsilon_0}} \quad (8)$$

where $\lambda(n)$ is a "forgetting factor," whose schedule is parameterized by γ, which controls the asymptotic decay rate of $\eta(n)$, and ϵ_0, which sets the initial length of the "memory window." Notice that $\eta(n)$ can be computed recursively:

$$\eta(n) = \frac{1}{1 + \frac{\lambda(n)}{\eta(n-1)}} \quad (9)$$

Thus, in this approach, the stepsize schedule is separated into three regions. In the range $1 < n < \frac{1}{\epsilon_0}$, a rough (but reliable) estimator is formed using a short memory window. In the second phase, $\frac{1}{\epsilon_0} < n < \frac{1}{\gamma\epsilon_0}$, learning is carried out with a window length of $\frac{1}{\epsilon_0}$ (i.e., $\eta(n) \approx \frac{1}{\epsilon_0}$). In the final phase, $\eta(n)$ decays as $\frac{\gamma+1}{\gamma n}$, resulting in the asymptotic results from stochastic approximation.

In our experiments on learning of speaker-dependent GMVQs for LSF quantization, we found that the values $\gamma = 0.05$ and $\epsilon_0 = 0.001$ resulted in reliable estimation. The learning curves for this problem, averaged over all 45 speakers, are seen in Figure 5. Note the small disparity between the likelihood performance and the actual quantization performance. This results from the assumption that the inertial profile of the quantizers is independent of the parameters, which is not really accurate. This is not a major impediment, in that the variation in the inertial profile is small compared to the changes in likelihood that are achieved.

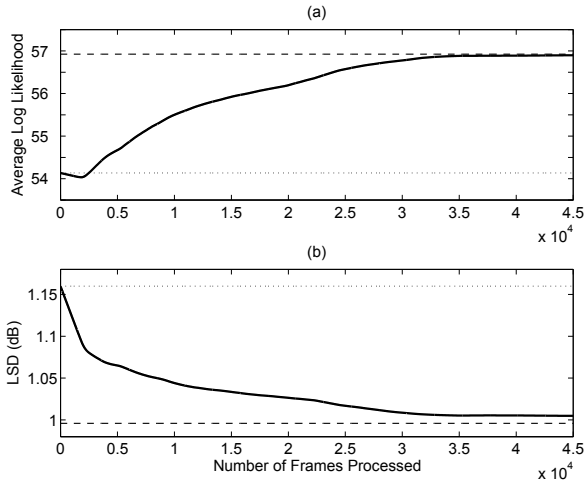

Fig. 5. Convergence of Online EM Algorithm. Subfigure (a) shows the average log likelihood while subfigure (b) depicts the average Log Spectral Distortion. The dashed lines show the performance of speaker-dependent systems trained using batch methods, while the dotted lines show speaker-independent performance. Note that the final performance achieved by the recursive procedure is within 1/4 of a bit per frame of the batch learning performance.

4. DISCUSSION

This paper has presented two techniques for use in online training of GMVQ systems. First, learning from quantized data was considered. This technique enables the learning process to be carried out at locations remote from the encoder (and, hence, the end-user's equipment). The key to learning GMVQ systems from data quantized by other GMVQ systems is to apply a post-processing to the decoded data which avoids the problem of quantized outputs lying in subspaces. The impact on the performance of the resulting GMVQs as a function of the encoding rate was quantified, and shown to be around 1 bit per frame for standard "transparent quality" operating rates. Next, recursive learning was considered. This learning technique replaces the standard batch EM approach to GMVQ learning with a frame-by-frame approach which has minimal storage requirements. It was demonstrated that a suitable choice of learning schedule results in performance equivalent to that obtained by batch EM methods, implying that there is effectively no penalty to using recursive learning. Taken together, these two techniques enable a wide variety of online training configurations, in that they eliminate the need to carry out the training in the end-user's equipment (by using quantized data) and remove onerous storage requirements associated with batch EM training.

5. REFERENCES

[1] A. D. Subramaniam and B. D. Rao, "PDF Optimized Parametric Vector Quantization of Speech Line Spectral Frequencies" IEEE Trans. on Speech and Audio Proc., Vol. 11, (no.2), March 2003.

[2] E. R. Duni and B. D. Rao, "Performance of Speaker-Dependent Wideband Speech Coding" INTERSPEECH 2007, Anwerp, Belgium.

[3] D.M. Titterington, "Recursive Parameter Estimation using Incomplete Data" J. R. Statist. Soc. B, vol. 46, no. 2, pp. 257-67, 1984.

[4] D.M. Titterington and J-M. Jiang, "Recursive Estimation Procedures for Missing-Data Problems" Biometrika, vol. 70, no. 3, pp. 613-24, 1983.

[5] M. Sato, "Fast Learning of On-Line EM Algorithm" Unpublished manuscript.

[6] J.-L. Gauvain and C.-H. Lee, "Maximum a Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains" IEEE Trans. on Speech & Audio Proc., vol. 2, no. 2, April 1994.

[7] E. R. Duni, "High-Rate Optimized Quantization Structures and Speaker-Dependent Wideband Speech Coding" Ph.D. Thesis, University of California, San Diego, 2007.

[8] P-J. Chung and J. F. Bohme, "Recursive EM and Sage-Inspired Algorithms With Application to DOA Estimation" IEEE Trans. on Signal Proc., Vol. 53, no. 8, Aug. 2005.