ADAPTATION OF COMPRESSED HMM PARAMETERS FOR RESOURCE-CONSTRAINED SPEECH RECOGNITION

Jinyu Li, Li Deng, Dong Yu, Jian Wu, Yifan Gong, and Alex Acero

Microsoft Corporation, One Microsoft Way, Redmond, WA 98052
jinyuli@ece.gatech.edu, {deng;dongyu;jianwu;ygong;alexac}@microsoft.com

ABSTRACT

Recently, we successfully developed and reported a new unsupervised online adaptation technique, which jointly compensates for additive and convolutive distortions with vector Taylor series (JAC/VTS), to adjust (uncompressed) HMMs under acoustically distorted environments [1]. In this paper, we extend that technique to adapt compressed HMMs using JAC/VTS where limited computation and/or memory resources are available for speech recognition (e.g., on mobile devices). Subspace coding (SSC) is developed and used to quantize each dimension of the multivariate Gaussians in the compressed HMMs. Three algorithmic design options are proposed and evaluated that combine SSC with JAC/VTS, where three different types of tradeoffs are made between recognition accuracy and the required computation/memory/storage resources. The strengths and weaknesses of these three options are discussed and shown on the Aurora2 task of noise-robust speech recognition. The first option greatly reduces the storage space and gives 93.2% accuracy, which is the same as the baseline accuracy but with little reduction in the run-time computation/memory cost. The second option reduces about 79.9% of the computation cost and about 33.5% of the memory requirement at a very small price of 0.5% decrease of accuracy (to 92.7%). The third option cuts about 89.2% of the computation cost and about 65.5% of the memory requirement while reducing recognition accuracy by 2.7% (to 90.5%).

Index Terms—resource constraint, subspace coding, joint compensation, additive and convolutive distortions, mobile devices

1. INTRODUCTION

With the rapid increase in the usage sceneries of mobile devices, it is natural to put automatic speech recognition (ASR) on mobile devices, which are ideal platforms for hands-free applications. Two critical factors affecting the applications on mobile devices are speed and memory requirements as they are always resource limited. Usually, speech recognition is designed running on machines with high computation power and large memory. To be used in mobile devices, ASR systems must be tailored to meet the computation and memory limits. A popular choice is to use vector quantization (VQ) and compress the HMM parameters. Different VQ methods have been proposed to quantize the Gaussian vectors or sub-vectors [2].

Differing from other applications in mobile devices, speech recognition faces the additional, special issue of noise robustness. Noise robustness in speech recognition remains an outstanding and difficult problem despite many years of research and investment. Developing noise robustness techniques for mobile applications is even harder. Because of the very nature of mobility, mobile devices can be used in any environments, often acoustically adverse. The speech signals input into mobile devices can be distorted by many possible types of distortions, including additive and convolutive distortions and their mixes, which are not easy to predict accurately during recognizers' development. Therefore, the actual models for the distorted speech deviate from the original models trained in advance. The originally trained models may be adapted to fit the environment with maximum likelihood linear regression (MLLR) [3] or maximum a posterior (MAP) [4]. However, to achieve acceptable performance the MLLR method often requires significantly more than one transformation matrix, and this inevitably results in demanding requirements for the amount of the adaptation data [5]. MAP requires even more data than MLLR.

It is desirable to have an online adaptation strategy because the mobile devices are used in different environments at different time. Joint compensation of additive and convolutive distortions (JAC) in the model domain appears to be a good choice. For each input speech utterance, JAC estimates the noise and channel distortion parameters of the utterance online and has achieved promising results [6][7][8], where the static part of HMM parameters is adapted. An extended framework (JAC/VTS) is developed and reported in [1] to adapt both static and dynamic parts of HMM parameters, and the same framework is used to estimate the static and dynamic parts of noise and channel mean and variance parameters with the help of vector Taylor series (VTS). This results in much better performance in noise robustness.

The research presented in this paper studies how to modify our JAC/VTS algorithm that was originally developed for the normal, uncompressed HMMs so that it can be effectively deployed in resource-limited devices where compressed HMMs are used. In Section 2, we outline our JAC/VTS algorithm for uncompressed HMM parameters and the implementation steps. In Section 3, a subspace coding (SSC) method is described for compressing HMM parameters and three strategies for combining SSC with JAC/VTS are discussed. Experimental evaluation of the three strategies is provided in Section 4. We summarize our study and draw conclusions in Section 5.

2. JAC/VTS ADAPTATION ALGORITHM

2.1 Algorithm for HMM Adaptation Given the Joint Noise and Channel Estimates

The first part of the overall JAC/VTS algorithm is to adjust or

adapt the original HMM parameters trained with clean speech. This is done using the estimates of the noise and channel parameters, which is the second part of the overall algorithm described in subsection 2.2.

In the distortion model that we use in this work, the observed distorted speech signal *y[m]* is generated from clean speech signal *x[m]* with noise *n[m]* and channel's impulse response *h[m]* according to $y[m] = x[m]*h[m] + n[m]$. (1)

For the given noise mean vector μ_n and channel mean vector μ_h (all in the MFCC domain), we define matrix $G(.)$ that depends on μ_x for the *k*-th Gaussian in the *j*-th state according to

$$G(j,k) = C\, diag\left(\frac{1}{1+\exp(\, C^{-1}(\mu_n-\mu_{x,jk}-\mu_h)\,)}\right)C^{-1}. \quad (2)$$

Then, the Gaussian mean vectors (the *k*-th Gaussian in the *j*-th state) in the adapted HMM for the degraded speech become

$$\mu_{y,jk} \approx \mu_{x,jk} + \mu_h + C\,\log(\,1+\exp(\, C^{-1}(\mu_n-\mu_{x,jk}-\mu_h)\,)\,). \quad (3)$$

The covariance matrix $\Sigma_{y,jk}$ in the adapted HMM is

$$\Sigma_{y,jk} \approx G(j,k)\Sigma_{x,jk}G(j,k)^T + (I-G(j,k))\Sigma_n(I-G(j,k))^T \quad (4)$$

For the delta and delta/delta portions of MFCC vectors, the adaptation formulas for the mean vector and covariance matrix are

$$\mu_{\Delta y,jk} \approx G(j,k)\mu_{\Delta x,jk} + (I-G(j,k))\mu_{\Delta n}, \quad (5)$$

$$\mu_{\Delta\Delta y,jk} \approx G(j,k)\mu_{\Delta\Delta x,jk} + (I-G(j,k))\mu_{\Delta\Delta n}, \quad (6)$$

$$\Sigma_{\Delta y,jk} \approx G(j,k)\Sigma_{\Delta x,jk}G(j,k)^T + (I-G(j,k))\Sigma_{\Delta n}(I-G(j,k))^T, \quad (7)$$

$$\Sigma_{\Delta\Delta y,jk} \approx G(j,k)\Sigma_{\Delta\Delta x,jk}G(j,k)^T + (I-G(j,k))\Sigma_{\Delta\Delta n}(I-G(j,k))^T \quad (8)$$

2.2 Algorithm for Re-estimation of Noise and Channel

EM algorithm is developed as the second part of the overall JAC/VTS algorithm to jointly estimate all the noise and channel parameters using the first-order VTS approximation. The re-estimation formulas for the static channel mean μ_h, the static and dynamic noise means [μ_n, $\mu_{\Delta n}$, $\mu_{\Delta\Delta n}$], and the static and dynamic noise variances [Σ_n, $\Sigma_{\Delta n}$, $\Sigma_{\Delta\Delta n}$] are given with detailed derivation in [1], and are used in the experiments reported in this paper.

3. JAC/VTS FOR COMPRESSED HMMS

The JAC/VTS adaptation algorithm described above cannot be applied directly for resource-constrained speech recognition (e.g., on mobile devices) because all of the very large number of Gaussians in the HMM system must be updated for each utterance. This requires a very large amount of computation as well as memory resources. How to modify the above JAC/VTS algorithm to meet the requirement of limited computation/memory resources is presented next.

In this section, a simple version of subspace coding (SSC) method is first introduced to compress the parameter of HMMs. Then, three architectural options for combining SSC with the JAC/VTS algorithm are given.

3.1 Subspace Coding for Compressing HMM Parameters

In the subspace coding (SSC) scheme that we have implemented and used in our experiments, for an HMM system consisting of *K* Gaussians each with fixed dimension *D* and with a diagonal covariance matrix, the quantization is performed along each separate dimension of the Gaussians. Dimension *d* of Gaussian *k* $(m_k[d], v_k^2[d])$ is quantized into a two-dimensional vector $(\mu_n[d], \sigma_n^2[d])$. The standard k-means clustering algorithm is used to design the codebook with a conventional distortion measure for Gaussian models. The centroid of a cluster is computed by uniform quantization.

3.2 Integration strategies for JAC/VTS with SSC

Option 1: This simplest architectural option for integrating JAC/VTS with SSC is appropriate when the storage space is the only concern for mobile devices. In this architecture, the original HMMs are compressed with SSC and stored in a potential mobile device. For each incoming utterance, the SSC-compressed HMMs are expanded as a full HMM set first, and then are adapted using JAC/VTS as described in Section 2.

Option 2: Practical scenarios usually do not allow all HMMs to be updated for each utterance. This would be time consuming and memory demanding. We hence modify Option 1 into Option 2 by modifying the HMM update block as shown in the right corner of Figure 1. In this option, in the final stage, only the models that have been "seen" (in the recognized transcription) during decoding are adapted. This greatly reduces the computation and memory requirement. After adapting HMMs, we also add a new block to update SSC code vectors with the "seen" models while keeping the original mapping relation in SSC, i.e., only $(\mu_n[d], \sigma_n^2[d])$ are changed. The code vector update uses two options: one is the uniform coding, and the other is data-driven coding with sample weights obtained from the statistics of the current speech utterance.

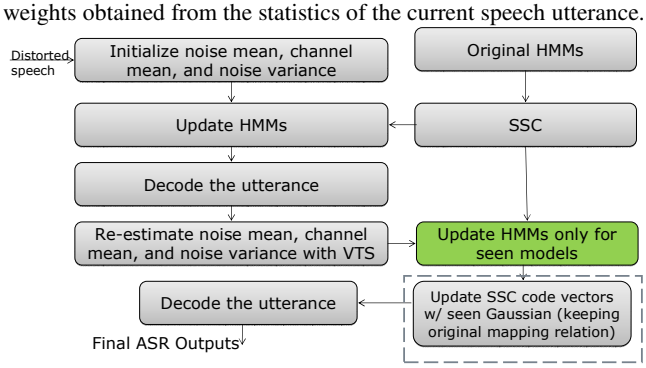

Figure 1: Architecture of Option 2 for JAC/VTS + SSC. Only the "seen" models are updated in the final adaptation stage.

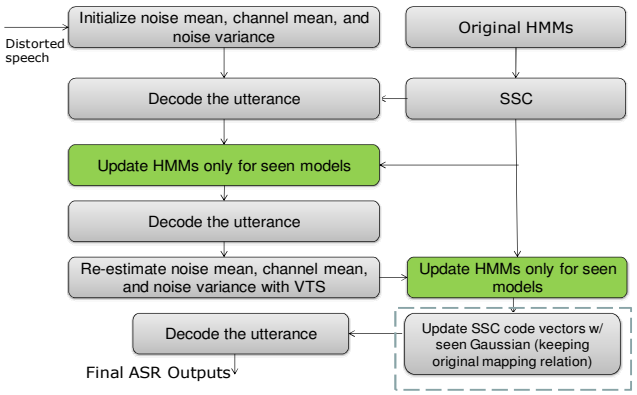

Figure 2: Architecture of Option 3 for JAC/VTS + SSC. Only the "seen" models are updated in both the adaptation stages.

Option 3: There are two modules for adapting HMMs. We only modify the adaptation of the final stage in Option 2; this is incomplete for computation and memory reduction. It is desirable to update all HMMs with "seen" models in both stages in Option 3. Unfortunately, we need to know what the "seen" models are in the first HMM update block. As a result, Option 3 requires one additional decoding step in order to provide the "seen" model list as shown in Figure 2.

4. EXPERIMENTS

The integration of the JAC/VTS algorithm with SSC as described in the preceding section was evaluated on the standard Aurora 2 task [9] of recognizing digit strings in noise and channel distorted environments. The clean and multi-style training sets are used to train the baseline maximum likelihood (ML) estimated HMMs. These models are denoted as clean-trained and multi-trained models. The test material consists of three sets of distorted utterances. The data in set-A and set-B contain eight different types of additive noise, while set-C contains two different types of noise plus additional channel distortion. The features are 13-dimension MFCCs, appended by their first- and second-order time derivatives. The cepstral coefficient of order zero is used instead of the log energy in the original script.

To show the effects of SSC, the standard complex "backend" of HMMs provided by ETSI is used throughout our experiments. There are 11 whole-digit HMMs, one for each of the 11 English digits, including the word "oh". Each HMM has 16 states, and each state is modeled by a Gaussian mixture model (GMM) with 20 Gaussians. In addition, there are one "sil" and one "sp" model. The "sil" model consists of 3 states, and each state is modeled by a GMM with 36 Gaussians. The "sp" model has only one state and is tied to the middle state of the "sil" model. This configuration gives a total of 3664 Gaussians.

SSC is used to compress the HMMs. For each dimension of Gaussian vectors, 256 clusters are used to represent the 3,664 Gaussians. Therefore, the compression ratio is about 14

The digit recognition accuracy results (averaged over Sets-A, -B, and -C, with the standard Aurora2 criterion) in Row one of Table 1 show the baseline system performance obtained using clean-trained and multistyle-trained HMMs, respectively. Without JAC/VTS, the clean-trained model obtains 50.6% accuracy (Acc) and the multi-trained model obtains 89.3% Acc. After applying SSC, the clean-trained model's accuracy drops slightly to 48.7% and multi-trained model drops dramatically to 83.2%. A possible reason for the latter is the use of uniform VQ in the codebook design.

Then, the integration strategy Option 1 is examined by applying JAC/VTS on the compressed models with the results shown in Row 2 of Table 1. The power of JAC/VTS is clearly demonstrated by improving the 48.7% Acc from the clear-trained model with SSC to 92.2% Acc. Although JAC/VTS was developed to work with the clean-trained model, performance improvement is also achieved for the multi-trained model with SSC by increasing Acc from 83.2% to 93.2%. There are two possible reasons for this. First, the multi-trained model gives better posteriors and better transcription for unsupervised adaptation than the clean-trained model. Second, since noise and channel parameters are trained automatically, breaking modeling assumptions by replacing the clean HMM with the multi-trained HMM may have been partly compensated for.

Option 1 may not be practical for resource-limited speech recognition without considering the computation and memory cost when expanding and adapting all Gaussians in the HMMs from the VQ codebook. In the remainder of this section, we will report the evaluation of Option 2 and Option 3 systems for combining JAC/VTS with SSC. (Several other options have been explored but will not be reported in this paper due to space limit.) These more practical options make tradeoffs between recognition accuracy and computation/memory resources, with the upper and lower bounds in the performance established in Table 1. That is, for the clean-trained model, the accuracy's lower bound is 48.7% and upper bound 92.2%. For the multi-trained model, the corresponding bounds are 83.2% and 93.2%, respectively.

Table 1: Accuracy results of the baseline system (Row 1 with no JAC/VTS) and the Option-1 system (Row 2 with integrated JAC/VTS and SSC)

Using JAC/VTS	Clean-trained	clean-trained +SSC	multi-trained	multi-trained +SSC
No	50.64%	48.65%	89.34%	83.21%
Yes	92.21%	92.21%	93.21%	93.21%

The integration strategy Option 2 updates only a subset of the HMMs from a "seen" model list, which are obtained from the recognition result. Table 2 compares the performance within Option 2 with different ways of using VQ. For the multi-trained model, if only the HMMs inside the recognition transcription are adapted in the final stage, saving the computation cost by about half, then the recognition accuracy drops from Option 1 by only 0.5% absolute with no use of VQ. For the clean-trained model, the corresponding accuracy drop is 1.8% absolute (Row 1). Then two types of VQ are used to compress the HMMs after adaptation, while keeping the original mapping relation in SSC. Data-driven VQ is shown to perform better than uniform VQ but still reduces recognition accuracy compared with using no VQ.

Table 2: Option-2 system performance (one best)

Update "Seen" HMMs in the Final Stage	clean-trained +SSC+JAC/VTS	multi-trained +SSC+JAC/VTS
no VQ	90.43%	92.69%
data-driven VQ	88.84%	91.83%
uniform VQ	87.03%	90.85%

Although satisfactory results are obtained, the Option-2 system still needs to adapt all Gaussians in the HMMs in the first stage, which may not be afforded in mobile devices. The Option-3 system adapts all Gaussians in the HMMs from the "seen" lists in both stages. The SSC-compressed original HMMs are used to decode the distorted speech to obtain the transcription serving as the "seen" model list for the first-stage HMM adaptation. Unfortunately, as shown in Table 3, this drops performance by a rather significant amount. For the multi-trained model, the accuracy is dropped to 87.8% (no VQ), and for the clean-trained model, to much lower accuracy of 67.5%. The main reason for the performance degradation is that HMM adaptation now strongly relies on the transcription provided by the original HMMs. For the clean-trained SSC-compressed model, the accuracy is as low as 48.7% to begin with. Therefore, the quality of the "seen" model list provided by the decoding transcription may be too poor to make adaptation effective. Again, similar to Option 2, the use of VQ

decreases the performance further, and data-driven VQ is also better than uniform VQ.

Table 3: Option-3 system performance (one-best)

Update "Seen" HMMs in All Stages	clean-trained +SSC+JAC/VTS	multi-trained +SSC+JAC/VTS
no VQ	67.47 %	87.81%
data-driven VQ	66.06%	86.52%
uniform VQ	64.77%	85.60%

To examine the extent to which the quality of transcription affects the recognition performance in the Option-3 system, we conducted an experiment that the "seen" model list in the first HMM adaptation stage is obtained from a 5-best list of decoding, instead of the top-1-best candidates (the latter shown in Table 3). With more models being "seen" and adapted, the recognition accuracy of the multi-trained model for the Option-3 system jumps from 87.8% to 90.5%. The corresponding increase for the clean-trained model is from 67.5% to 72.8% (Row 2 in Table 4). With an increasing number of candidates with selecting larger N in the N-best list, more and more HMMs will be adapted and the performance will approach that reported in Table 2. The optimal selection will depend on the performance tradeoff with the computational/memory resource allowed in actual deployment.

Table 4: Option-3 system performance (one-best vs. 5-best)

Update "Seen" Models in All Stages	clean-trained +SSC+JAC/VTS	multi-trained +SSC+JAC/VTS
1-best	67.47 %	87.81%
5-best	72.75%	90.49%

Here we give a very coarse analysis of the computation and memory costs of the three options presented above. Since most of the resources are allocated by JAC/VTS rather than decoding, the analysis is conducted with the analysis for JAC/VTS only.

Option 1 compresses the HMMs with a compression scale of 14. Hence, it reduces the storage space with a factor of 14. Because Option 1 adjusts all the HMMs, it does not reduce the memory requirement. In the first stage, because the HMMs are shared with code vectors, the computation cost is only 1/14 of that of the uncompressed model. Therefore, the total computation cost reduction is 100%*(1-(1/14 + 1)/2) = 46.4%.

For Option 2, in the first JAC/VTS stage all HMMs (100%) are adapted and in the second stage only the "seen" HMMs (roughly 1/3 in our experiment) are adapted. Thus, the relative decrease of the memory requirement is 100%-(100%+ 33%)/2 = 33.5%. The computation cost reduction is 100%-(100%/14+ 33%)/2 = 79.9%, since the models in the first stage are compressed and 1/3 models in the second stage are adapted.

For Option 3, in the first stage, the HMMs in the 5-best list (roughly 51%) are adapted (49% models are un-adapted and still compressed); and in the second stage only the "seen" HMMs are adapted. Hence, the relative memory reduction is 100%-(51%+ 33%*(51%+49%/14))/2 = 65.5% . The reduction of computation is 100%-(51%/14+ 33% * (51%+49%/14))/2 = 89.2%.

5. SUMMARY AND CONCLUSION

In this paper, we have presented our recent study aimed to integrate our earlier successful JAC/VTS algorithm into the HMMs compressed by SSC for model adaptation on resource-limited devices. The goal of this study is to explore the feasibility of noise-robust ASR in mobile devices with computation, memory, and storage constraints.

Three integration options are proposed and evaluated that save, to a varying degree, computation cost, and memory, and/or storage compared with the straightforward JAC/VTS on the regular, uncompressed HMMs. Evaluated on the Aurora2 task, the original clean-trained and multi-trained models obtained 92.2% and 93.2% accuracy after being adapted by JAC/VTS. The SSC-compressed models achieve the same accuracy level with Option 1, which saves storage by a factor of 14 but not computation and memory. In Option 2, recognition accuracy drops somewhat to 90.4% (clean-trained) and 92.7% (multi-trained), with nearly half computational/memory cost. In the most compact Option 3, the accuracy drops further to 72.8% (clean-trained) and 90.5% (multi-trained), given a 5-best list in the first decoding stage. This performance degradation is attributed mainly to the unsupervised nature of JAC/VTS.

Three immediate research issues will be addressed in our future work. First, the quality of compressed HMMs needs improvement so as to overcome the difficulty caused by the unsupervised nature of JAC/VTS. Since the current, initial study reported in this paper is focused on how to combine SSC with JAC/VTS, little effort has been devoted to the quality of SSC. Second, confusion patterns will be used to replace the *N*-best list to provide "seen" model lists for adapting selected HMMs. This will not only reduce the computation requirement but also improve the "seen" list quality, especially for clean-trained models. Third, the JAC/VTS algorithm will be further improved to provide greater effectiveness of HMM adaptation [1].

6. REFERENCES

[1] J. Li, L. Deng, D. Yu, Y. Gong, and A. Acero, "A unified framework of HMM adaptation with joint compensation of additive and convolutive distortions," submitted to *IEEE Trans. Audio, Speech, and Language Proc.*, 2007.

[2] E. Bocchieri and B. K. -W. Mak, "Subspace distribution clustering hidden Markov model," *IEEE Trans. Speech and Audio Proc.*, vol. 9, no. 3, pp. 264-275, 2001.

[3] C. J. Leggetter and P. C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density HMMs," *Comput., Speech, Lang,*, vol. 9, no. 2, pp. 171–185, 1995.

[4] J. -L. Gauvain and C. -H. Lee, "Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains," *IEEE Trans. Speech Audio Processing*, vol. 2, pp. 291–298, 1994.

[5] X. Cui and A. Alwan, "Noise robust speech recognition using feature compensation based on polynomial regression of utterance SNR," *IEEE Trans. Speech and Audio Proc.*, vol. 13, pp. 1161-1172, 2005.

[6] Y. Gong, "A method of joint compensation of additive and convolutive distortions for speaker-independent speech recognition," *IEEE Trans. Speech and Audio Proc.*, vol. 13, no. 5, pp. 975-983, 2005.

[7] D. Y. Kim, C. K. Un, and N. S. Kim, "Speech recognition in noisy environments using first order vector Taylor series," *Speech Communication*, vol. 24, pp. 39-49, 1998.

[8] P. Moreno. *Speech Recognition in Noisy Environments*. PhD. Thesis, Carnegie Mellon University, 1996.

[9] H. G. Hirsch and D. Pearce, "The Aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions," *Proc. ISCA ITRW ASR*, 2000.