Automatic Construction of Regression Class Tree for MLLR via Model-based Hierarchical Clustering

Shih-Sian Cheng [1, 2], Yeong-Yuh Xu [1], Hsin-Min Wang [2], and Hsin-Chia Fu [1]

[1] Department of Computer Science, National Chiao-Tung University, Hsinchu
{yyxu, hcfu}@csie.nctu.edu.tw
[2] Institute of Information Science, Academia Sinica , Taipei
{sscheng, whm}@iis.sinica.edu.tw

Abstract. In this paper, we propose a model-based hierarchical clustering algorithm that automatically builds a regression class tree for the well-known speaker adaptation technique - Maximum Likelihood Linear Regression (MLLR). When building a regression class tree, the mean vectors of the Gaussian components of the model set of a speaker independent CDHMM-based speech recognition system are collected as the input data for clustering. The proposed algorithm comprises two stages. First, the input data (i.e., all the Gaussian mean vectors of the CDHMMs) is iteratively partitioned by a divisive hierarchical clustering strategy, and the Bayesian Information Criterion (BIC) is applied to determine the number of clusters (i.e., the base classes of the regression class tree). Then, the regression class tree is built by iteratively merging these base clusters using an agglomerative hierarchical clustering strategy, which also uses BIC as the merging criterion. We evaluated the proposed regression class tree construction algorithm on a Mandarin Chinese continuous speech recognition task. Compared to the regression class tree implementation in HTK, the proposed algorithm is more effective in building the regression class tree and can determine the number of regression classes automatically.

Keywords: speaker adaptation, MLLR, regression class tree

1 Introduction

MLLR [1] is well known for its ability to perform rapid and robust speaker adaptation with a small amount of adaptation data. Extensive research efforts have been made to improve MLLR [8, 13] as well as to develop new methods that extend the conventional MLLR framework [2-7].

In the MLLR proposed by Leggetter and Woodland [1], adaptation of speaker independent (SI) model parameters (e.g., the mean parameters of a CDHMM-based speech recognition system) is carried out via a set of linear transformations, where each regression (transformation) matrix is responsible for the adaptation of one regression class (subset of the model parameters). To enhance flexibility and robustness, the authors proposed using of a regression class tree to group the parameters of the model set into regression classes. The purpose is to dynamically determine the sharing of regression matrices for the parameters according to the amount and type of adaptation data available [8]. The regression class tree is a critical component in the MLLR framework as well as in other linear transformation based approaches, e.g., [3].

The issue of regression class tree construction for MLLR can be viewed as a data clustering problem of the parameters. For example, HTK [9] applies a centroid splitting algorithm to construct a regression class tree, in which the number of base clusters (classes) must be determined empirically. In this study, we developed a model-based hierarchical clustering algorithm, which not only provides a better clustering result for the model parameters, but also determines the number of clusters (i.e., base classes of the regression class tree) automatically. The proposed regression class tree construction algorithm is a two-stage process. In the first stage, the input data is iteratively partitioned in a top-down fashion using a divisive hierarchical clustering strategy, and the Bayesian Information Criterion (BIC) [10] is applied to determine the number of clusters. In the second stage, these clusters are iteratively merged in a bottom-up fashion to build the regression class tree. To evaluate the performance, the proposed regression class tree implementation was compared with that of HTK. The experimental results show that the proposed algorithm is effective in building a regression class tree automatically and in determining the number of regression classes for MLLR.

The rest of this paper is organized as follows. First, MLLR and the concept of regression class tree are reviewed in Section 2. Then, the proposed algorithm for regression class tree construction is introduced in Section 3. The experimental results are presented in Section 4, followed by our conclusions in Section 5.

2 MLLR and regression class tree

In MLLR, to adapt the SI Gaussian mean vectors for example, the mean vectors are clustered into C regression classes, and each regression class c is associated with an $n\times(n+1)$ regression matrix $\mathbf{W}_c$, where n is the dimensionality of the feature vector. Let the mean vector $\boldsymbol{\mu}_m = [\mu_m(1),\ldots, \mu_m(n)]^{\mathrm{T}}$ of Gaussian component m be one of the T_c mean vectors in the regression class c; then, the adapted mean vector can be derived as

$$\hat{\boldsymbol{\mu}}_m = \mathbf{W}_c \xi_m = \mathbf{A}_c \boldsymbol{\mu}_m + \mathbf{b}_c, m = 1,\ldots,T_c; c = 1,2,\ldots,C, \quad \textbf{(1)}$$

where $\xi_m=[1, \mu_m(1),\ldots, \mu_m(n)]^{\mathrm{T}}$ is the $(n+1)$-dimensional augmented mean vector. $\mathbf{A}_c$ and $\mathbf{b}_c$ are an $n\times n$ matrix and an n-dimensional vector, respectively, such that $\mathbf{W}_c = [\mathbf{b}_c \ \mathbf{A}_c]$. $\mathbf{b}_c$ is used as a bias vector. $\mathbf{A}_c$ can be diagonal, block-diagonal, or full. $\{\mathbf{W}_c\}_{c=1,\ldots,C}$ is estimated by maximizing the likelihood of the adaptation data for the adapted parameters using EM algorithm.

To facilitate flexibility and robustness, MLLR usually makes use of a regression class tree. All the Gaussian components are arranged into a tree, which is basically a binary tree, such that close components in the acoustic space are grouped in the same node (regression class). The lower level of the tree indicates that the components are more close. In the hierarchy of the tree, each parent node contains all the components of its two child nodes, and all the leaf nodes are termed as *base classes*. During the adaptation process, the feature vectors used for adaptation are aligned to the corresponding Gaussian components, and the occupation counts are accumulated for each of the base classes. The regression class tree can be traversed in either a top-down or a bottom-up fashion to only generate transformations for those nodes that have sufficient adaptation data. Fig. 1 shows an example of a regression class tree. The numbers in italics associated with the tree nodes are the number of adaptation feature vectors aligned to them. If the threshold for the sufficiency of the adaptation data is set as 300, only the transformations for regression nodes 2, 3, and 4 will be constructed. The transformation of node 2 will take charge of the adaptation of Gaussian components in node 5, and the transformation of node 3 will take charge of nodes 6 and 7.

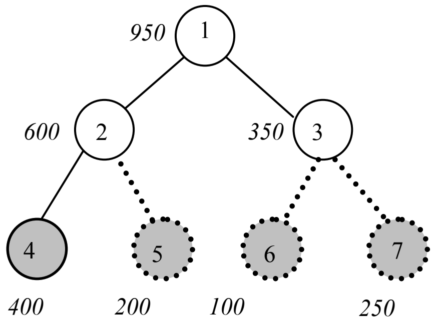


Fig. 1. An example of a regression class tree.

3 Model-based hierarchical clustering for automatic regression class tree construction

In this section, before describing the proposed regression class tree construction algorithm in detail, we briefly introduce BIC, which provides the splitting and merging criteria for the proposed algorithm.

3.1 Model selection and BIC

Given a data set $X=\{x_1, x_2,\ldots, x_n\}$ and a set of candidate models $M=\{M_1, M_2,\ldots, M_k\}$, the model selection problem is to choose the model that best fits the distribution of X. BIC is a model selection criterion and the BIC value of model M_i is

$$BIC(M_i, X) = \log p(X \mid \hat{\Theta}_i) - \frac{1}{2}\#(M_i)\log n, \qquad \textbf{(2)}$$

where $p(X \mid \hat{\Theta}_i)$ is the maximum likelihood of X for model M_i, and $\#(M_i)$ is the number of parameters of M_i. The model with the highest BIC value is selected. The

BIC-based approach is also known as a penalized likelihood approach, which gives a larger penalty to more complex models.

3.2 The proposed regression class tree construction algorithm

The proposed regression class tree construction algorithm is a two-stage process. In the first stage, the input data X is viewed as a single cluster initially, after which the clusters are divided into finer clusters iteratively by using BIC as the validity criterion for splitting until there is no cluster should be split. Then, in the second stage, similar to agglomerative hierarchical clustering, these clusters are iteratively merged in a bottom-up fashion to build the resultant dendrogram. The details of the proposed clustering algorithm are given in Algorithm 1, which we call TDBU (Top-Down & Bottom-Up). There are two major issues with respect to the proposed clustering algorithm:

(***I1***) In the Top-Down (TD) stage, which cluster should be split into a pair of sub-clusters and how should it be split?
(***I2***) In the Bottom-Up (BU) stage, what is the appropriate distance measure of two clusters and how should they be merged?

On Issue (***I1***).
At each splitting iteration, each cluster C_i with $\Delta BIC_{21}(C_i)=BIC(GMM_2, C_i) - BIC(GMM_1, C_i)$ larger than 0 is split into two sub-clusters, where GMM_k represents a Gaussian mixture model with k mixture components. According to BIC theory, the larger the value of $\Delta BIC_{21}(C_i)$, the better GMM_2 will fit C_i, and thus the more confidence there will be that C_i is composed of at least two Gaussian clusters. As to the splitting of cluster C_i, after the training of GMM_2, each sample belonging to C_i is distributed to the Gaussian component that has the largest posterior probability for the sample. In other words, suppose Θ_1 and Θ_2 are the two components of GMM_2, for each x in C_i, then x is distributed to $cluster_{\Theta j}$ if $j=\arg\max_r p(\Theta_r|x)$.

On issue (***I2***).
At each merging iteration in the second stage, the two most similar (close) clusters are merged into a single cluster. Given two clusters, C_i and C_j, let $C'=\{ C_i , C_j \}$. Then, $\Delta BIC_{21}(C')$ is used to represent the dissimilarity (or distance) between C_i and C_j. The smaller the $\Delta BIC_{21}(C')$ value the more confident we are in describing the distribution of C' as one Gaussian cluster.

In the proposed TDBU algorithm, the TD stage alone can construct a regression class tree. However, the regression class tree constructed by the following BU stage is believed to be better than that constructed by the TD stage alone. The TD stage can capture the real clusters in X approximately, but may not construct an optimal dendrogram for the real clusters because of the uncertainties of the splitting processes and the suboptimal hierarchy construction of the clusters. We consider that the major contribution of the TD stage is to automatically determine the number of clusters in X and to provide a decent clustering result for the BU stage to start with.

After the TD stage, the BU stage can construct a better hierarchy for these clusters, since it proceeds as the conventional (non-model-based) hierarchical agglomerative clustering. Fig. 2 illustrates the clustering process of the TDBU algorithm with a simple example. We can clearly see the differences between the dendrograms constructed by the TD stage alone and by the complete TDBU process. The memory complexity of the BU stage for storing the distance matrix is $O(m^2)$, where m is the number of clusters produced by the TD stage, compared to $O(n^2)$ for the conventional hierarchical agglomerative clustering approach, where n is the number of input samples. Obviously, $O(m^2)$ is smaller than $O(n^2)$.

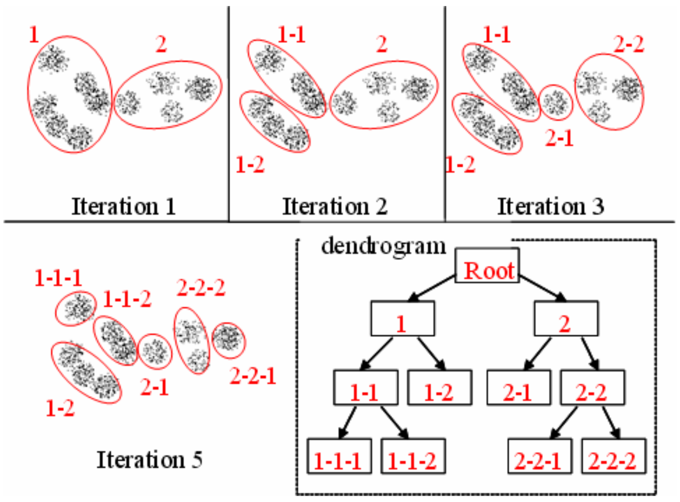
```
Algorithm: TDBU
Input: Data set  $X=\{x_1, x_2,\dots, x_n\}$.
Output: A dendrogram of the input data set $X$.
Begin
  Top-Down (TD) stage:
    1. Start with one single cluster (the root node of the TD
       dendrogram).
    2. Repeat:
         Split cluster (leaf node) $C_i$ with $\Delta BIC_{21}(C_i)>0$ into
         two new clusters (leaf nodes).
       Until there is no cluster (leaf node) whose $\Delta BIC_{21}$
       value is larger than 0.
  Bottom-Up (BU) stage:
    1. Start with the resultant clusters $C_1, C_2,\dots, C_m$ in the
       TD stage (the leaf nodes of TD dendrogram).
    2. Repeat:
         Merge the two closest clusters (nodes) into a single
         cluster (parent node) at the next level of the BU
         dendrogram.
       Until only one cluster (root node) left.
    3. Output the BU dendrogram.
End
```

Algorithm 1. The proposed model-based hierarchical clustering algorithm for MLLR regression class tree construction.

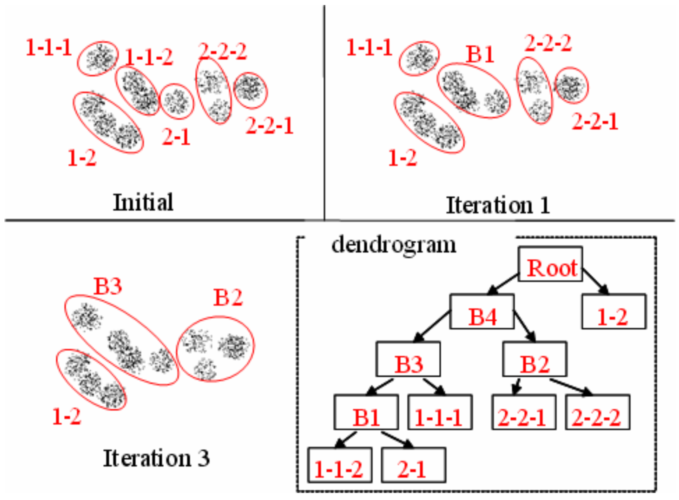
4 Experiments

4.1 Experimental setup

The proposed approach was evaluated on the TCC300 continuous Mandarin Chinese microphone speech database [12], which contains data of 150 female and 150 male speakers. The speech data of 260 speakers, a total of 23.16 hours was used to train the



(a) TD stage



(b) BU stage

Fig. 2. An example of the TDBU clustering process. The resultant clusters at iteration 5 of the TD stage are fed to the BU stage as the initial condition. The dendrogram constructed in the BU stage is the output of TDBU.

SI acoustic model, while the speech data of eight speakers (four female and four male), not included in the 260 training speakers was used for model adaptation and testing. The sampling rate of the speech was 16 kHz. Twelve MFCCs and log-energy, along with their first and second order time derivatives, were combined to form a 39-dimensional feature vector. Utterance-based Cepstral mean subtraction (CMS) was applied to the training and test speech to remove the channel effect.

Considering the monosyllabic structure of the Chinese language in which each syllable can be decomposed into an INITIAL/FINAL format, the acoustic units used in our speech recognizer are intra-syllable right-context-dependent INITIAL/FINAL, including 112 context-dependent INITIALs and 38 context-independent FINALs [11]. Each INITIAL is represented by a CDHMM with three states, while each FINAL is represented with four states. The number of Gaussian components for each state is 32. For each test speaker, about 125 seconds of speech data was used for model adaptation, while 400 seconds was used for speech recognition evaluation. In the adaptation experiments, the 125-second adaptation speech for each test speaker was averagely chopped into 25 five-second utterances. The recognizer performed only free syllable decoding without any grammar constraints. Syllable accuracy was used as the evaluation metric. All adaptation experiments were conducted in a supervised manner and only mean vectors of Gaussian components in the SI model were adapted. The speaker independent recognition accuracy was 66.20%, averaged over the eight test speakers. The performance of the built-in approach in HTK [9] was used as the baseline result. The speaker adaptation experiments on the proposed approach were also performed with HTK.

4.2 Experimental results

Fig. 3 shows the adaptation performance of various regression class trees constructed by the built-in HTK approach and the proposed algorithm - TDBU. The number of base classes predefined for HTK ranged from 4 (denoted as HTK4) to 200 (denoted as HTK200). Full-covariance Gaussians were used to compute the ΔBIC value in the TDBU approach, and the number of base classes automatically determined by TDBU was 34. For each test speaker, the 25 five-second utterances were used for adaptation in order. For example, if the number of utterances is five, the adaptation was performed on the first five utterances.

Several conclusions can be drawn from Fig. 3: (1) When the amount of adaptation data is small (less than 10 utterances), there is no significant difference between the performance of all the approaches tested due to the very limited adaptation data. (2) If more adaptation data (more than 10 utterances) is available, the performance can be improved with more complex regression class trees (more base classes). 34 seems to be an appropriate number of base classes since the performance of HTK34, HTK64, and HTK200 is almost the same and are superior to the results obtained with fewer base classes. (3) It is clear that TDBU34 outperforms HTK34, HTK64, and HTK200. The experiment results show that the TDBU approach is not only more effective than the regression class tree implementation method in HTK, but can also find an appropriate number of base classes automatically during the regression class tree construction process. This is an advantage when we need to take account of the memory requirement of the regression class tree when designing an embedded speech recognition system for a device with limited memory.

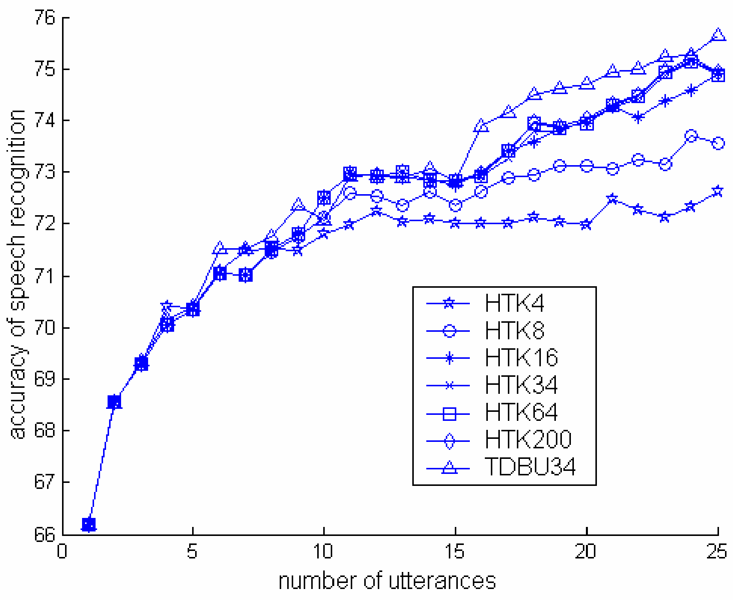


Fig. 3. Adaptation performance obtained with various regression class trees constructed by HTK and TDBU. The number of base classes determined by TDBU is 34.

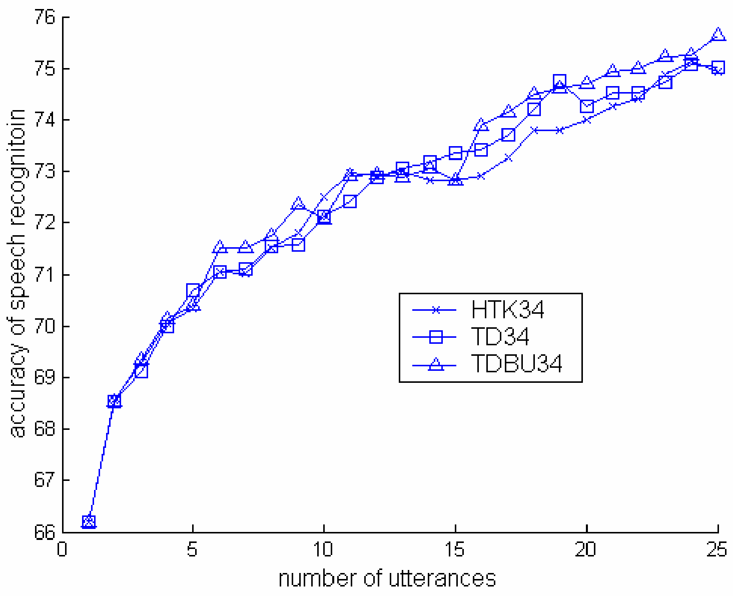


Fig. 4. Adaptation performance of HTK34, TD34 and TDBU34.

As mentioned in Section 3, the TD stage (i.e., the first stage of TDBU) can be used alone to construct the regression class tree. Fig. 4 depicts the performance curves of TD34, TDBU34 and HTK34, from which we can infer that performing the BU stage after the TD stage definitely constructs a better hierarchy for the regression classes than that constructed using the TD stage alone. The experiment results also show that, in general, the TD34 regression class tree outperforms the HTK34 regression class tree.

5 Conclusion

This paper presents a model-based hierarchical clustering algorithm for MLLR regression class tree construction. The experiment results shows that the regression class tree constructed by our approach is more effective than that constructed by HTK. In addition, our approach can automatically decide an appropriate number of regression classes, which used to be decided empirically.

References

1. Leggetter, C. J. and Woodland, P. C.: Maximum Likelihood Linear Regression for Speaker Adaptation of Continuous Density Hidden Markov Models. Computer Speech and Language, vol. 9 (1995) 171-185.
2. Chesta, C., Siohan, O., and Lee, C.-H.: Maximum a Posteriori Linear Regression for Hidden Markov Model Adaptation. Proc. EUROSPEECH'1999.
3. Siohan, O., Myrvoll, T.-A., and Lee, C.-H.: Structural Maximum a Posteriori linear Regression for Fast HMM Adaptation. Workshop on Automatic Speech Recognition 2000. ISCA ITRW ASR'2000.
4. Chen, K. T., Liau, W. W., Wang, H. M., and Lee, L. S.: Fast Speaker Adaptation Using Eigenspace-based Maximum Likelihood Linear Regression. Proc. ICSLP'2000.
5. Chen, K. T. and Wang, H. M.: Eigenspace-based Maximum a Posteriori Linear Regression for Rapid Speaker Adaptation. Proc. ICASSP'2001.
6. Doumpiotis, V. and Deng, Y.: Eigenspace-based MLLR with Speaker Adaptive Training in Large Vocabulary Conversation Speech Recognition. Proc. ICASSP'2004.
7. Mak. B. and Hsiao. R.: Improving Eigenspace-based MLLR Adaptation by Kernel PCA. Proc. ICSLP'2004.
8. Leggetter, C. J. and Woodland, P.C.: Flexible Speaker Adaptation Using Maximum Likelihood Linear Regression. Proc. ARPA Spoken Language Systems Technology Workshop, 1995.
9. HTK Speech Recognition Toolkit, http://htk.eng.cam.ac.uk/
10. Fraley, C. and Raftery, A. E.: How Many Clusters? Which Clustering Method? Answers via Model-based Cluster Analysis. Computer Journal, 41 (1998) 578-588.
11. Wang, H. M. et al.: Complete Recognition of Continuous Mandarin Speech for Chinese Language with Very Large Vocabulary Using Limited Training Data. IEEE Trans. on Speech and Audio Proc., 5(2) (1997) 195-200.
12. The Association for Computational Linguistics and Chinese Language Processing, http://www.aclclp.org.tw/corp.php
13. Gales, M. J. F.: Maximum Likelihood Linear Transformations for HMM-based Speech Recognition. Computer Speech and Language, vol. 12 (1998) 75-98.