

Implementation of Regression Class Tree based CMLLR in the TKK speech recognizer

Peter Smit

Department of Information and Computer Science
Aalto University, School of Technology

September 30, 2010

Abstract

Regression Class Tree based CMLLR is implemented in the TKK recognizer and experiments are done to validate the results. Comparison with the old system shows that the CMLLR part of the code is working correctly. A study of the likelihood of the adaptation data shows that, as expected, it increases together with the number of adaptation matrices. Recognition performance however, does not always increase.

1 Introduction

In the summer of 2009 I was working in the Speech Group of the Department of Information and Computer Science of the Helsinki University of Technology as a summer trainee. My assignment during the summer traineeship was to make an implementation of Regression Class Tree Based CMLLR as described in Gales [1996] for the internal TKK speech recognizer. This report records the work and findings for the course T-61.5910 Research Project in Computer and Information Science.

The implementation of the CMLLR is done as described in Gales [1998]. For the implementation details, the Hidden Markov Toolkit (HTK) and the already existing global CMLLR implementation in the TKK recognizer are examined.

2 Theory

2.1 ASR models

There are generally two types of automatic speech recognition (ASR) models. Speaker Independent (SI) models are trained with data from multiple different speakers and are expected to work reasonably well for a big range of speakers. On the other hand, Speaker Dependent (SD) models are trained specifically for one speaker, resulting in a better performance for this speaker compared to a SI model. Other speakers however, perform in general poorly with this model. In addition, a SD model needs lots of data and lots of training for every single speaker, therefore making it infeasible for use in generic systems.

The solution for this would be to derive a SD model from a SI model with use of an adaption technique. An adaptation require much less data and training time than a SD model, while still giving good performance for a single speaker like in a SD model.

2.2 HMM for Acoustic Modelling

The common way of modelling the acoustic signals are Hidden Markov Models (HMM). HMM's are an extension of a normal markov chain, where every output is generated by a hidden input state. Through inference (often done with the Viterbi algorithm), the sequence of hidden states can be found given a sequence of outputs. When modelling speech the hidden states in the model represent the phonemes (or context dependent phonemes like tri-phones) and the output distributions are generating MFCC vectors; the preprocessed speech.

The output distribution is often a multivariate Gaussian distribution or Gaussian Mixture Model (GMM). Commonly, diagonal covariance matrices are used, under the assumption that the elements in the MFCC are independent and to relax the computational requirements for the calculation of probabilities. Except for the transition probabilities between states, all parameters of the HMM are defined by the output distributions.

2.3 Maximum Likelihood Linear Regression

One popular adaptation technique is Maximum Likelihood Linear Regression (Leggetter and Woodland [1995]), which adapts the mean and optionally the variance of every distribution by applying a linear transformation. The transformation is estimated so that the Likelihood of the input speech samples of one speaker is maximized.

The transform of the mean $\boldsymbol{\mu}$ with transformation matrix \mathbf{A} and bias \mathbf{b} to obtain the new mean $\hat{\boldsymbol{\mu}}$ is given by:

$$\hat{\boldsymbol{\mu}} = \mathbf{A}\boldsymbol{\mu} - \mathbf{b} \quad (1)$$

The transformation matrix and bias are calculated with the Expectation-Maximisation algorithm.

When in MLLR the variance is transformed, it is done independently from the mean.

2.3.1 Constrained Maximum Likelihood Linear Regression (CMLLR)

To reduce the number of parameters of the transform, the mean and variance can be adapted with the same transform using Constrained Maximum Linear Likelihood Regression (CMLLR) as described in Digalakis et al. [1995] and Gales [1998]. In CMLLR the mean $\boldsymbol{\mu}$ and the variance $\boldsymbol{\Sigma}$ are transformed with the following equations:

$$\hat{\boldsymbol{\mu}} = \mathbf{A}^{-1}\boldsymbol{\mu} - \mathbf{b}^{-1} \quad (2)$$

$$\hat{\boldsymbol{\Sigma}} = \mathbf{A}^{-1}\boldsymbol{\Sigma}\mathbf{A}^{-1T} \quad (3)$$

The Maximum Likelihood solution given some speech data can be estimated with an iterative solution as shown in Gales [1998]. For the speech data the following statistics are collected:

$$\mathbf{G}^{(i)} = \sum_{m=1}^M \frac{1}{\sigma_i^{(m)2}} \sum_{\tau=1}^T \gamma^{(m)}(\tau) \zeta(\tau) \zeta(\tau)^T \quad (4)$$

$$\mathbf{k}^{(i)} = \sum_{m=1}^M \frac{1}{\sigma_i^{(m)2}} \mu_i^{(m)} \sum_{\tau=1}^T \gamma^{(m)}(\tau) \zeta(\tau)^T \quad (5)$$

Now the rows of the extended transformation matrix $\mathbf{W} = [\mathbf{b}^T \quad \mathbf{A}^T]^T$ can be update iteratively by

$$\mathbf{w}_i = \left(\alpha \mathbf{p}_i + \mathbf{k}^{(i)} \right) \mathbf{G}^{(i)} - 1 \quad (6)$$

where \mathbf{p} is the extended cofactor row vector $[0, c_{i1}, \dots, c_{in}]$ of \mathbf{A} , with $c_{ij} = \text{cof}(\mathbf{A}_{ij})$. The cofactor ij of a matrix is the determinant of this matrix with the row i and column j removed. All the cofactor can be together calculated by taking the diagonal elements of $|\mathbf{A}| \mathbf{A}^{-1}$.

2.4 Regression Class Trees

In the standard MLLR and CMLLR methods, all the Gaussian distributions are transformed with the same transformation. Nevertheless, if there is enough data, it might be beneficial to divide the models output distributions in regression classes and to calculate a separate transform for each regression class.

Regression Class Trees are giving a framework to generate possible regression classes and to dynamically determine the number of classes at recognition time (Gales [1996]). The creation of a tree is start with a node containing all different output distributions and then working down by splitting nodes based on a splitting measure. Because Gaussian Mixture distributions are used, the measure used is the Euclidean distance.

For the procedure of building the tree the gaussians can all be used separately, or be grouped by mixture or phoneme. Grouping reduces the amount of units in the tree and already keeps similar gaussians together. A reduced amount of units makes it easier to generate the tree, but could prevent the tree from expanding very deeply (as there can be only as much terminal nodes as units).

An example of a regression tree with 5 terminal nodes is shown in Figure 1.

2.4.1 Regression Class Tree Generation

The process of generating a Regression Class Tree is an iterative process. Besides the units: gaussians, mixtures or phonemes; and their occupancy in the training data, the only parameter is the maximum desired amount of terminal nodes in the generated tree.

The algorithm starts with placing all units in the root node. Then, iteratively the terminal node with the highest score is split until the desired number of terminal nodes are created. Off course, this process is stopped earlier if every terminal nodes exists of only one unit, and therefore can not be split.

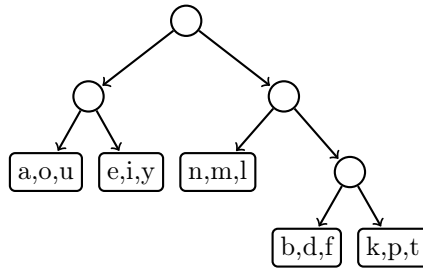


Figure 1: An example regression tree with 5 terminal nodes

The score of a node is the Euclidean distance between all Gaussians in the node and their mean, weighted by their occupancy in the training data. When splitting the node, the sum of the scores of the two new nodes will be the minimization criterium. This procedure is similar to the K-means algorithm for $K = 2$, except that the weighted distances are used. In short, first the group of units is split in two and the weighted means of the groups are calculated. Secondly, every unit is reassigned to a mean and the means are recalculated. The process is repeated until convergence, and no units are changing group anymore.

2.4.2 Regression Class Tree Usage

When the Regression Class Tree is used in recognition another parameter is needed; the minimum occupancy of a transformation matrix. If a transformation matrix has too little data to be estimated robustly, it will not generalize well to unseen data, effectively degrading ASR performance.

The process of determining the units that for this transformation are grouped together works in the following way. First all the needed statistics ($\mathbf{G}^{(i)}$ and $\mathbf{k}^{(i)}$) are collected from the adaptation data, separately for every terminal node. After the process is completed, the occupancy of every terminal node is checked. If not enough data is collected, the node is merged with its sibling to form a new terminal node. It can easily be done as the statistics can be summed together. In the case that the sibling is not a terminal node, first all its children are merged together in the same fashion.

As a result the tree has only terminal nodes with enough data for the estimation of a transformation.

3 Implementation

The TKK recognizer, workname aku, is the internal recognizer of the Speech Group in the Computer Science and Engineering Department of Aalto University. The recognizer is developed in C++ by the TKK speech group and used in assisting the research done on large vocabulary ASR systems. The recognizer is specialized for recognizing Finnish, a morphological rich language with a very straightforward and consistent pronunciation.

Before the project, the recognizer was already able to do (single-matrix) CMLLR adaptation, which was implemented as a feature transformation for

the MFCC vector used in the acoustic model. The code for estimating the transformations was specific for the one matrix case.

Besides that the concept of Regression Classes was not existent and needed to be implemented, also the way of applying the transformation needed to be reformed, because the transformation depends on which regression class the distribution is in.

3.1 Tree generation

The most challenging part in tree generation is to be able to work with different units. Therefore, a template based solution was chosen and implemented for the tree types of units. The training was fairly trivial with the help of the existing libraries for Matrix calculations.

3.2 Transform Estimation

For the estimation of the transform, the code was fully rewritten. Besides for allowing multiple Regression Classes to be trained, also computational cost: memory and time; were taken into account. As a result, the final solution was performing in the case of a single matrix at least as fast as the original code.

3.3 Applying of the transforms in recognition

The way of using the transforms in a computationally smart manner was the biggest challenge of the implementation. Because the original approach in the feature extraction part of the code could not be used any more, due to the dependency on the model, a new architecture had to be developed.

One of the goals was to leave the code of the Acoustic model as intact as possible and to not break any existing functionalities in the core of the recognizer.

There are two approaches for applying the transformation. The first one is to transform the distributions, which however has serious computationally drawbacks. This has to be done for every Gaussian in the system and makes the covariance matrices of the distribution full instead of diagonal, impeding the performance of the core task.

The second and used approach was to wrap the interfaces of the Gaussian classes. Every Gaussian in the system is replaced by a fake class that acts as a proxy to the original Gaussian class. However, every time the probability of a vector is tested, the vector is transformed by it's appropriate matrix. Because the transformation is the same for all Gaussians in the same Regression class, this transformation is cached and reused. This leads to only the inexpensive task of wrapping the Gaussians and only calculating the transformed feature vector for every transformation once.

4 Experiments

Several experiments were done to test the implementation. Firstly, tests were performed to verify the correctness of the single matrix adaptation. This was

Baseline	Unsupervised CMLLR
3.3	2.8

Table 1: Letter Error Rate (LER) for Speecon Evaluation set

tested by comparing the results of the new version with the already existing implementation in the old version.

The tree based CMLLR was verified by looking to the likelihood of training samples. In this case more transformation matrices should, for the same data, always give an increase in likelihood.

At last, normal recognition experiments are done and the results interpreted.

4.1 Single transformation experiments

The single transformation experiment was executed with the common setup of the TKK recognizer for the Finnish Speecon corpus as also used in Pylkkönen [2009]. The Speecon Evaluation set has 40 speakers with on average 28 sentences per speaker. Both the baseline and single CMLLR adaptation were tested. The adaptation was unsupervised and all speaker data in the test set was used for estimating the transforms.

In Table 1 the result for these two experiments are shown. The recognition result for both the original CMLLR implementation and the new implementation were exactly the same, so the CMLLR implementation proved to be successful.

4.2 Likelihood experiments

As the target of CMLLR is to increase the likelihood of the speech data, we expect to see an increase in (log) likelihood when we use more transformations. In Figure 2 the development of the total likelihood of the Speecon Evaluation data is shown when the number of adaptation matrices is increased. Note that this figure does not have a scale on the Log Likelihood axis, as the values are heavily dependent on the speech data and model and therefore not relevant.

The figure verifies the correctness in case of multiple classes. Every increase of the number of regression classes is increasing the likelihood of the adapted speech. Also as expected, it goes up steeply for a low number of classes and increases slowly when there are already a big number of classes.

4.3 Recognition experiments

The implementation is applied to a complete ASR experiment to test the improvement in recognition rate. Because the goal of an ASR system is to give the best possible performance, it is an important metric for the usefulness of multiple adaptation matrices. On the other hand, there are a lot of factors in the whole ASR system and particularities in the data that can influence the result, thus a positive result is not necessary for a specific dataset and model.

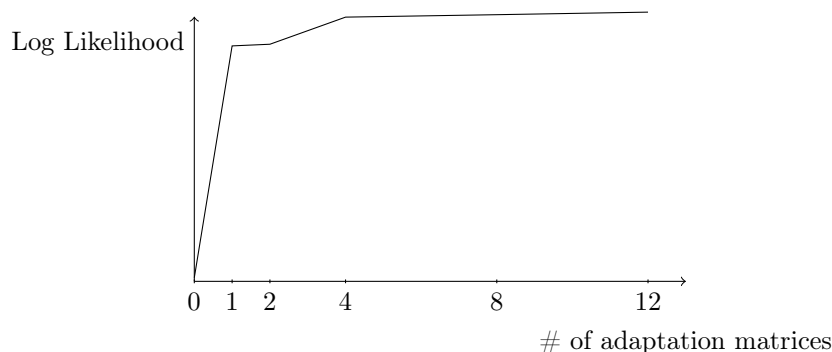


Figure 2: Log likelihood development when number of adaptation matrices is increased

4.3.1 Experiments on Finnish Speecon data

Experiments are performed on the Speecon corpus to test the performance of the model most used with the TKK recognizer. The standard TKK Speecon Evaluation set is used, and tests are performed with different number of adaptation matrices. The tree is build with phoneme units, but the results with different units were almost the same.

The results of this experiment are shown in Figure 3.

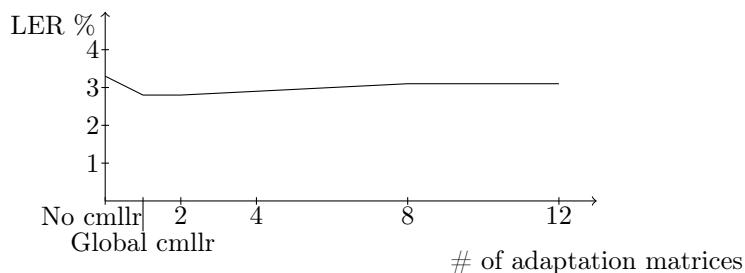


Figure 3: The Letter Error Rate for the Speecon Evaluation set with a different number of adaptation matrices. 0 matrices equals to no adaptation and 1 matrix to global CMLLR adaptation

Clearly multiple regression classes are not beneficial for the speecon model. The Letter Error Rate (LER) is approximately the same for a small number of matrices and gets worse for more matrices. The result is disappointing as at least some performance increase was expected.

4.3.2 Experiments on English Phone data

In Mansikkaniemi [2010] the developed Regression Class Tree based CMLLR in the TKK recognizer is put into use for English Law data. The results of this are shown in Figure 4. For this data a clear improvement in Word Error Rate (WER) is seen when multiple Regression Classes are used, compared to single-matrix CMLLR.

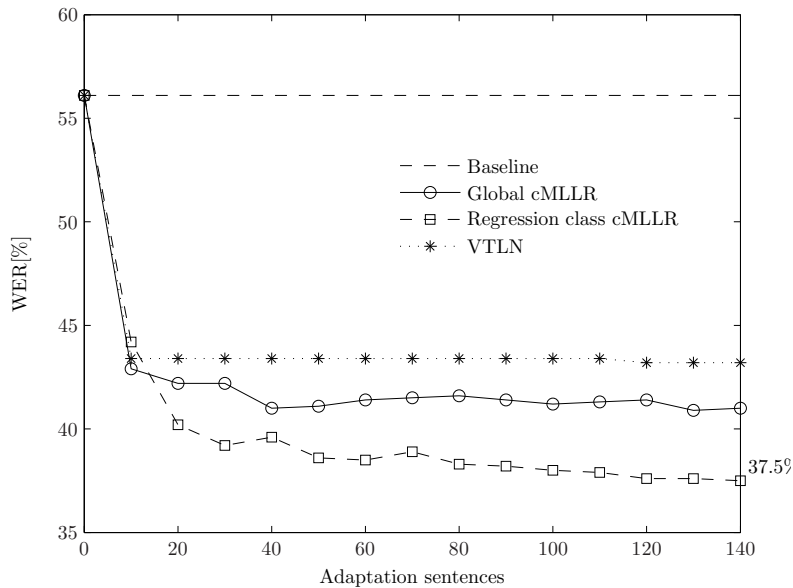


Figure 4: From Mansikkaniemi [2010], comparison of global CMLLR and Regression Class CMLLR for different amounts of adaptation data

5 Conclusion

This report described the implementation of Regression Class Tree based CMLLR in the TKK recognizer. Experiments have shown that the implementation performs as expected, in increasing the likelihood of the adaptation data. In recognition experiments however, a good result is not always obtained.

Further study could investigate why the Speecon experiment did not gain any improvement with Regression Class Tree CMLLR, while it is known for other recognizers to be beneficial.

Furthermore, another possible extension is to implement other transformation techniques with the implemented architecture, or to support multiple model-transformation in a single recognition (also called parent-transforms).

References

- V. Digalakis, D. Rtischev, and L. Neumeyer. Fast speaker adaptation using constrained estimation of Gaussian mixtures. *IEEE Trans. on Speech and Audio Processing*, pages 357–366, 1995.
- M.J.F. Gales. *The generation and use of regression class trees for MLLR adaptation*. Citeseer, 1996.
- M.J.F. Gales. Maximum likelihood linear transformations for HMM-based speech recognition. *Computer Speech and Language*, 12:75–98, 1998.
- C.J. Leggetter and P.C. Woodland. Maximum likelihood linear regression for

speaker adaptation of continuous density hidden Markov models. *Computer speech and language*, 9(2):171, 1995.

André Mansikkaniemi. Acoustic Model and Language Model Adaptation for a Mobile Dictation Service. Master's thesis, Aalto University, School of Technology, 2010.

J. Pykkönen. Investigations on discriminative training in large scale acoustic model estimation. In *Proceedings of the 10th Annual Conference of the International Speech Communication Association*, 2009.