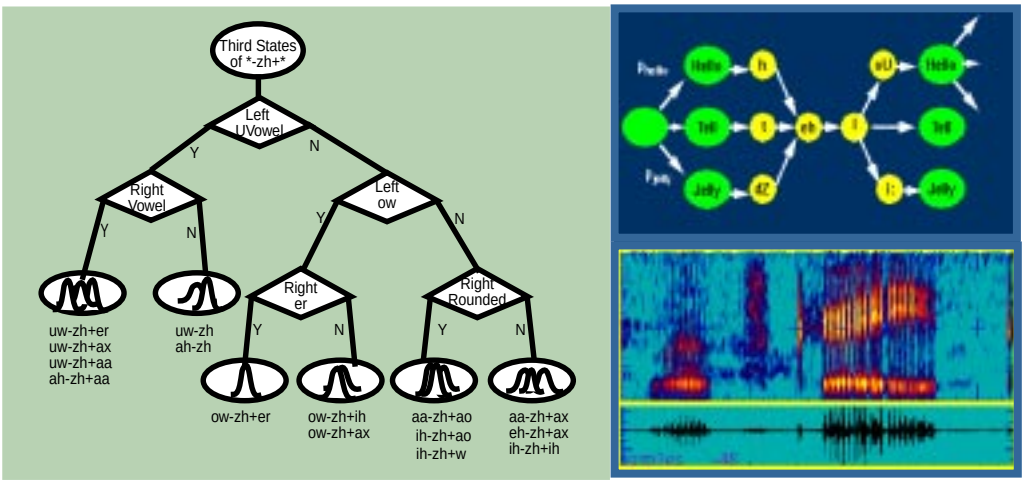
tutorial for

Decision Tree-Based State Tying For Acoustic Modeling

June 15, 1999



submitted by:

J. Zhao, X. Zhang, A. Ganapathiraju, N. Deshmukh, and J. Picone

Institute for Signal and Information Processing
Department of Electrical and Computer Engineering
Mississippi State University
Box 9571, 413 Simrall, Hardy Road
Mississippi State, Mississippi 39762
Tel: 662-325-3149, Fax: 662-325-3149
primary email contacts: {zhao, zhang, ganapath}@isip.msstate.edu

EXECUTIVE SUMMARY

Most state-of-the-art Large Vocabulary Conversational Speech Recognition (LVCSR) systems use context-dependent Hidden Markov Models (HMMs) to model speech data. In order to model the variations in speaker characteristics and pronunciations, it is common for an LVCSR system to comprise of several million parameters, which need to be estimated using several hours of speech training data. This explosion in the number of parameters is primarily because of the need to model acoustic units in terms of their context. However, many acoustic contexts are not observed with sufficient frequency in the training data, and therefore estimating model parameters for each context-dependent acoustic unit is difficult. Using discrete or semi-continuous density HMMs [2], or continuous density HMMs with tied parameters can significantly reduce the total number of model parameters to be estimated.

With continuous density HMMs which are popularly used in state-of-the-art LVCSR systems, phonetic state tying and mixture tying are used to reduce the total number of HMM parameters while maintaining the model accuracy. These can be implemented with traditional data-driven techniques such as k-means clustering, or knowledge-driven techniques like statistical decision trees. Decision tree-based state tying has gained popularity in recent years with successful application to several speech recognition tasks with a wide variety of complexity [1, 8-10]. The decision tree-based state tying algorithm uses both the training data as well as phonetically derived questions to cluster the states. It is also capable of handling models with contexts that rarely occur in the training data, if at all.

A similar decision tree-based module for phonetic state tying is now integrated in the ISIP STT toolkit. The implementation of this algorithm is based on the Maximum Likelihood (ME) principle, where the trees are grown till a significant increase in likelihood can be achieved. The likelihood computation is based on state occupancy counts produced during HMM training. The current implementation uses occupancy counts based on the Viterbi estimation algorithm.

The state tying module has two operating modes:

1. Training mode: During training, the decision trees are constructed in a top-down fashion by iteratively splitting the leaf nodes using phonetic questions and state occupancy counts. The terminal nodes of the tree represent the tied states.
2. Testing mode: The tree is used to generate models with unseen contexts. The result of this mode is a set of models corresponding to an user-specified list of context-dependent phones, as well as a list of clustered models.

The current implementation of the state tying module is specific to models with single Gaussian mixture components. It also assumes all models to have an equal number of states. We are currently in the process of using this model to train a context-dependent triphone system for Switchboard. As all software generated at ISIP, this module is implemented in object-oriented C++. The software and documentation are available in the public domain and can be accessed from *http://www.isip.msstate.edu/projects/speech/*.

TABLE OF CONTENTS

1. ABSTRACT

In Large Vocabulary Conversational Speech Recognition (LVCSR) system, the dominantly used acoustic models are Hidden Markov Models (HMMs). These models are trained to represent the acoustic units, such as phones (or usually, context-dependent phones). Because of the variations of sounds and pronunciations in speech, the number of context-dependent models is very large. If the parameters in those models are all distinct, the total number of model parameters would be very huge. It is common for an LVCSR system to have several million model parameters that need to be trained. Therefore parameter tying techniques are usually used to reduce the number of parameter, hence reduce the computation complexity. In this work, we implemented a decision tree-based state tying module, which uses both training data and phonetic knowledge to cluster states.

2. INTRODUCTION

The performance of a LVCSR system using HMM (Hidden Markov Models) depends on the accuracy of HMMs. It is expected that the more detailed models can more accurately represent the speech data. While since a typical LVCSR system requires thousands of models to account for all possible sounds in all possible contexts, the total number of parameters in these models is prohibitively large. The computation complexity to train all these parameters would be intolerable. On the other hand, the detailed models can accurately represent the speech data only if they are well trained by sufficient amount of training data. More parameters require more training data. Some specific model contexts may occur only a few times in the training data, or don't occur at all. One will never provide sufficient training data to train all the detailed models. Hence it's not necessary to keep all the parameters, since they are not guaranteed to be well trained.

To reduce the total number of model parameters, one approach is to reduce the number of parameters in each model, the other is to reduce the total number of distinct models or parameters. Using discrete HMMs or semi-continuous HMMs [2] belongs to the former approach. The advantage of this kind of approaches is the speed. The drawback is the discrete and semi-continuous HMMs are not as accurate as the continuous HMMs. Using continuous HMMs with tied parameters — parameter tying, belongs to the latter approach. It reduces the parameter count while maintaining the model accuracy, therefore it is popularly used in most state-of-the-art LVCSR systems. Parameter tying includes tying the whole models or tying part of modes such as state tying [1] and mixture tying [3].

Parameter tying is a standard clustering problem in pattern recognition. It can use algorithms such as k-means and decision tree algorithm. The clustering can start from both top-down and bottom-up. The scheme we used in this work is a decision tree-based top-down state tying technique, which has become increasingly popular in HMM-based LVCSR systems. We chose the decision tree algorithm, because it is both data-driven and knowledge-driven, it can handle the problem of unseen models. Unseen models refer to those models whose contexts occur only a few times or don't occur in the training data. In the phonetic decision tree, each node is associated with a set of states. These states are iteratively separated into child nodes by using phonetic questions. The states in the same leaf node share the same distribution, i.e. these states are tied.

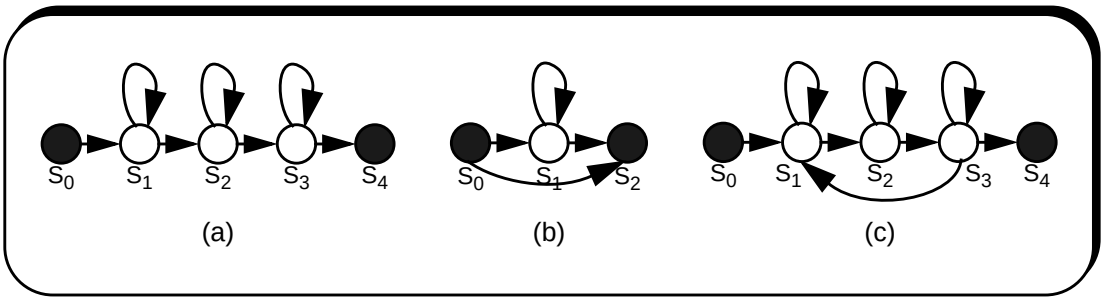


Figure 1. Some typical HMM topologies used for acoustic modeling in large vocabulary speech recognition — a) typical triphone b) short pause c) silence. The shaded states denote the start and stop states for each model

Once the decision trees are constructed, they can be used to estimate any logical HMMs. It doesn't matter whether the contexts represented by these models have occurred in the training data or not.

3. ACOUSTIC MODELS

In a speech recognition system, the analog speech signal is first converted into a sequence of feature vectors by the acoustic front-end. The acoustic models are used to implement phonetic classification. They need to provide a probability, or likelihood, of a word sequence to generate such feature vectors, or observations. It is impractical to do this calculation for every possible word in a large vocabulary application since it would require too many models to be processed. Hence, the word sequences are decomposed into basic sound units called phones.

3.1. Hidden Markov Models

A Hidden Markov Model (HMM) is used to model each phone (or usually, a context-dependent phone). An HMM is a doubly stochastic state machine that has a Markov distribution associated with transitions across various states, and a probability density function that models the output for every state. Depending on the complexity of the recognition problem, this distribution can be modeled as a discrete-valued or continuous-valued process. In speech recognition applications the choice of this output probability function is crucial as it must model all of the intrinsic spectral variability of real speech. Most current state-of-the-art systems use a mixture of multivariate Gaussian distributions to model context-dependent sequences of three phones (triphone models).

A Gaussian or a mixture of the continuous Gaussian probability density function (PDF) is:

$$Pr(x_t;\mu;C) = \frac{1}{\sqrt{(2\pi)^n|C|}} \cdot \exp\left(-\frac{1}{2}(x_t-\mu)^T \cdot C^{-1} \cdot (x_t-\mu)\right) \quad (1)$$

where x_t is the observation at time t, C and μ are the covariance matrix and mean vector of the Gaussian distribution. n is the order of the Gaussian distribution. Equation (1) gives the

probability of a state with the Gaussian distribution to generate x_t at time t.

Figure 1 shows some topologies of the HMM typically used to model context-dependent phones in LVCSR systems. Training an HMM to represent such a phone is to estimate the appropriate parameters such as the state transition matrix and the PDF's for each state. HMMs can be efficiently trained using the Baum-Welch forward-backward training algorithm [4] or the Viterbi algorithm [5]. As mentioned earlier, in a state-of-the-art LVCSR system using context dependent HMMs, there are possibly several million parameters that need to be trained. A popular way to reduce the parameter count while maintaining the accuracy of the models is the parameter tying.

3.2. Parameter Tying

Tying can be made at different levels. Two different but kind of similar models can share the same model or share some model parameters such as states or mixtures. There are two kinds of tying approaches. One kind is the top-down approaches, and the other kind is the bottom-up approaches.

Top-down approaches start by less distinct contexts, and then split them into more specific contexts. This is a standard classification problem. We can use classification techniques such as k-means and decision tree. A decision tree-based top-down approach uses phonetic knowledge together with the training data to decide which contexts are acoustically similar. The advantage of this kind of approaches is once the decision trees are constructed, they can be used to estimate unseen models. k-means is another commonly used classification technique. It is a data-driven technique, hence for the parameter tying problem, it can not handle unseen models as the decision tree algorithm does.

Bottom-up approaches on the contrary start by assuming that all contexts are distinct, and then merge the similar models to share parameters. These approaches use the distance between distributions to decide if two contexts are acoustically similar. The drawback of this kind of approaches is they require examples of each context to produce the initial estimates of the model parameter. For unseen models, usually it has to back-off to less specific models. In [6], a probabilistic mapping method is proposed.

State tying is HMM parameter tying at the state level. It allows some states in different models to share the same distribution, so the total number of distinct states in all the HMMs is reduced. Compared with tying the whole models, state tying makes finer distinctions between models by allowing left and right contexts to be modelled separately. If the distribution of a state is composed of multi-mixtures, the tying can be made one step further, that is the mixture tying. Mixture tying allows some Gaussian mixtures in different states to share the same distribution. Both state tying and mixture tying can be implemented by decision tree algorithm [1, 6-10].

4. DECISION TREE

Decision tree is a top-down approach mentioned above. It is a commonly used information-based classification technique.

4.1. Theory

A general decision tree consists of nodes, including non-leaf and leaf node. Each leaf node denotes a class. The input data consists of values of the different attributes. Initially, all these values are put inside the root node. By asking questions about the attributes, the decision tree splits the values into different nodes. Constructing a decision tree needs splitting rules and stopping rules. Once the decision tree is constructed, it can be used to evaluate other values to decide which classes they belong to.

The decision tree technique can be applied into the acoustic model state tying. In the phonetic decision tree, the values are a set of states in HMMs. The questions are some phonetic questions about the contexts. Each node in the tree contains a set of states, and there is a likelihood of these states generating the observation x. According to their answers to the question, the states can be separated into the left or right child node. For each child node, there is a new likelihood to generate x. The sum of these two child likelihoods should not be equal to the parent likelihood. The decision tree splitting rule is to maximize the likelihood increase after splitting. The stopping rules are some thresholds, such as the likelihood increase and the minimum number of state occupancy at each node. Figure 2 shows an example of the phonetic decision tree for the third state of triphones with the central phone “zh”. By asking questions, states which have similar

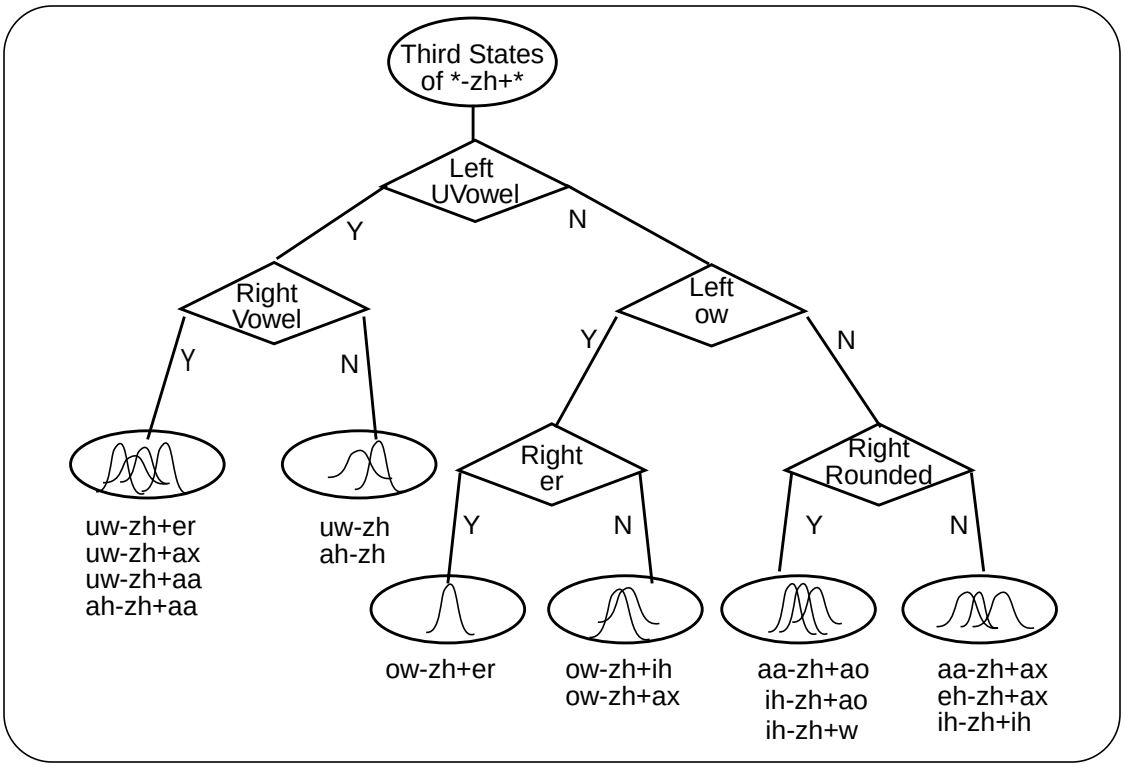


Figure 2. A phonetic decision tree for the third state of triphones with the central phone “zh”. Note: this tree was generated by the real algorithm. States at the leaf nodes are selected from the new models file.

acoustic contexts enter into the same leaf node and are tied together. For example, from Figure 2, it can be seen that the third states of “uw-zh+er”, “uw-zh+aa” and “ah-zh+aa” are tied together. because their left contexts are all UVowel, and right contexts are Vowel. They are acoustically similar, hence it makes sense to allow them share the same distribution.

The aim of splitting the phonetic decision tree is to maximize the likelihood of the tree. So computing the likelihood of each node is crucial in constructing the trees.

4.2. Likelihood Computation

The total likelihood of a tree to generate the observation x is the sum of the likelihood of each leaf node. Let L denote the likelihood of a node, which is approximated by the log probability of the distribution of this node to generate x weighted by its state occupancy. State occupancy is the occurrence count of a state.

$$L = \sum_t \left[\ln(Pr(x_t;\mu;C)) \cdot \sum_s \Upsilon_s(t) \right] \tag{2}$$

where s is a state at this node; x_t is the observation x at time t; $\Upsilon_s(t)$ is the state occupancy of s at time t and is a scalar. Since the distribution of every state is a Gaussian, the sum of Gaussian distributions is still a Gaussian distribution, which can be viewed as the distribution of the tied state of all the states at this node. Thus $\sum_s \Upsilon_s(t)$ is the occupancy at time t of the tied state. Let C and μ denote the covariance matrix and mean vector of the tied state. According to Equation (1),

$$\begin{aligned} \ln(Pr(x_t;\mu;C)) &= \ln\left(\frac{1}{\sqrt{(2\pi)^n |C|}} \exp\left(-\frac{1}{2}(x_t-\mu)^T C^{-1}(x_t-\mu) \right) \right) \\ &= \ln\left(\frac{1}{\sqrt{(2\pi)^n |C|}} \right) - \frac{1}{2}(x_t-\mu)^T C^{-1}(x_t-\mu) \\ &= -\frac{1}{2}\left\{ n\ln(2\pi) + \ln(|C|) + (x_t-\mu)^T C^{-1}(x_t-\mu) \right\} \end{aligned} \tag{3}$$

The covariance matrix can be given by:

$$C = \frac{\sum_t \left\{ \left(\sum_s \Upsilon_s(t) \right) \cdot (x_t-\mu)(x_t-\mu)^T \right\}}{\sum_t \sum_s \Upsilon_s(t)} \tag{4}$$

From (4), it can be derived that [see Appendix B].

$$\sum_t \left[(x_t - \mu)^T \cdot C^{-1} \cdot (x_t - \mu) \cdot \sum_s \Upsilon_s(t) \right] = n \cdot \sum_t \sum_s \Upsilon_s(t) \tag{5}$$

Hence, from equation (2), (3) and (5), we got the likelihood

$$L = \sum_t \left\{ -\frac{1}{2} [n \ln(2\pi) + \ln(|C|) + n] \cdot \sum_s \Upsilon_s(t) \right\} \tag{6}$$

$$= -\frac{1}{2} [n(1 + \ln(2\pi)) + \ln(|C|)] \cdot \sum_s \sum_t \Upsilon_s(t)$$

The sum of state occupancy over time $\sum_t \Upsilon_s(t)$ can be gotten from previous Viterbi training, the order of the sum of $\Upsilon_s(t)$ over s and t doesn't matter because the occupancy is a scalar. Covariance C of the sum distribution at this node can also be calculated by

$$C = E(x^2) - E(x)^2 = \frac{\sum_s [\Upsilon_s (C_s + \mu_s \mu_s^T)]}{\sum_s \Upsilon_s} - \left(\frac{\sum_s \Upsilon_s \mu_s}{\sum_s \Upsilon_s} \right) \left(\frac{\sum_s \Upsilon_s \mu_s}{\sum_s \Upsilon_s} \right)^T \tag{7}$$

where C_s and μ_s are the covariance matrix and mean vector of state s, and are known from the input untied states; Υ_s denotes $\sum_t \Upsilon_s(t)$, which is also known from the input.

Equation (6) and (7) are the two core equations used in the decision tree state tying algorithm.

4.3. Implementation

Figure 2 shows the flow of a general decision tree algorithm. There are two operation modes: train and test. The decision tree algorithm for state tying also follows this flow. For each particular state in models with the same center phone, one decision tree is constructed. For example, for 3 state triphone HMMs, if there are 80 monophones, 240 decision trees will be constructed.

In the train mode, at the beginning, this decision tree only has a root node. The states contained in this root node have a common attribute that they are at the same position in the models with the same central phone. Next, it needs to find the best split to split these states into child nodes. For each leaf node, there exists an optimal question, by using which to split the node, the likelihood increase is the maximum.

The likelihood increase is: $L = L(parent) - L(leftchild) - L(rightchild)$

So for each leaf node, first, the algorithm loops though all the questions, computes the likelihood increase, and finds the optimal question for that node. Then among all these leaf nodes, it chooses the one by splitting which to give the maximum likelihood increase. If the likelihood increase and the state occupancy after split exceed the thresholds, split this node. Repeat above steps until no split meets the thresholds. The states that result in the same leaf node are the tied states, which have the same state index and the same distribution.

In the test mode, given a model to be estimated, the algorithm first finds its decision trees corresponding to each state. Then by asking the optimal question associated with the node, the state enters into a child node until reaching a leaf node. So this state will be assigned the same state index represented by that leaf node.

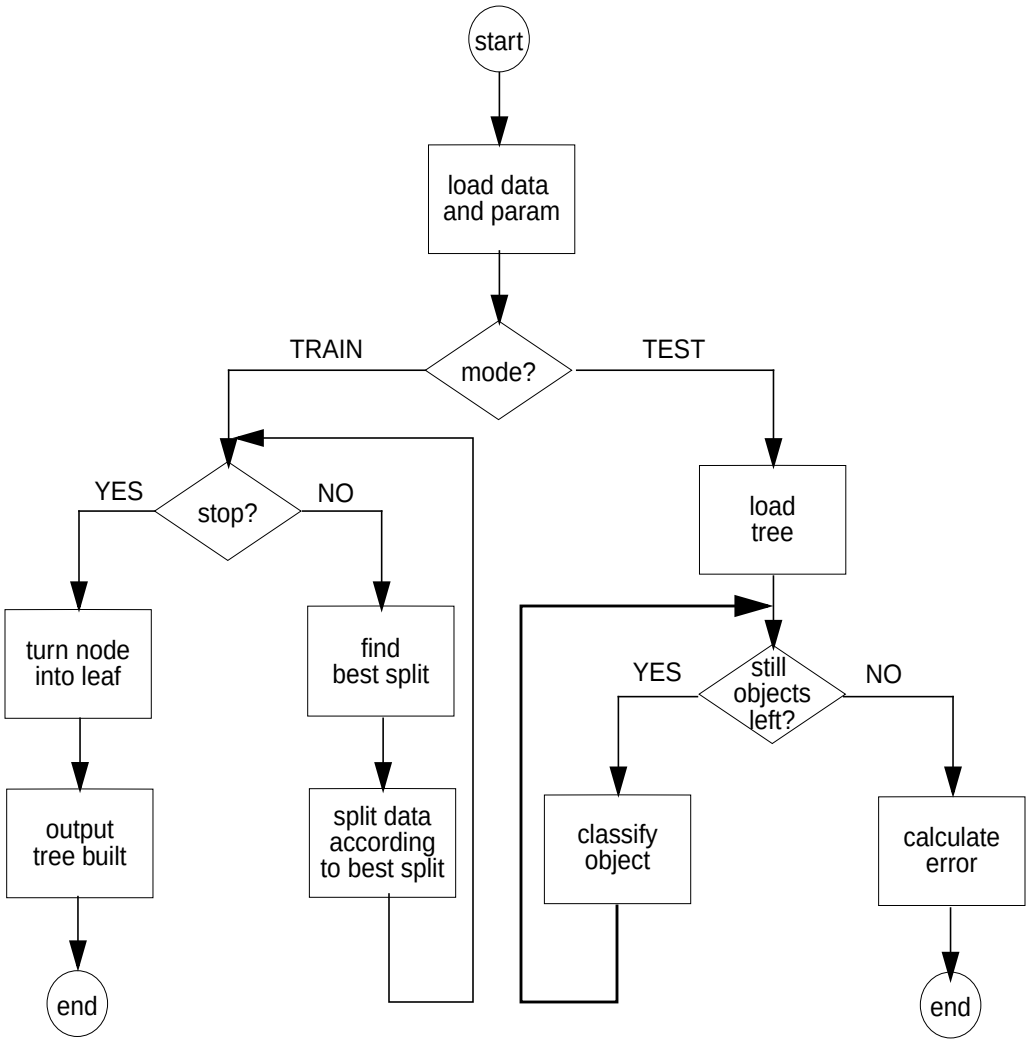


Figure 3. A general decision tree algorithm

4.4. Software Overview

This decision tree state tying algorithm was implemented in C++ using public domain GNU compiler. Four special classes — Decision_tree, Dt_node, Cd_model and Question, are constructed for this module. The utility of state tying has two operation modes. One is the train mode which reads in HMMs, untied states, state occupancies and questions etc., and outputs the tied states, the models with the updated state indices, and the decision trees. In the test mode, it reads in the decision trees, the list of phones to be estimated etc. The outputs are the models for these phones and the clist file which shows which phones share the same model.

The parameters file of the state tying utility is listed in Appendix A. The software and documentation are freely available at [11].

5. SUMMARY

We have implemented a decision tree-based state tying module for large vocabulary speech recognition system. It enables some states of different HMMs with similar contexts to share the same distribution. The evaluation of the tied states and new models is currently in progress.

6. FUTURE WORK

First, we will finish the evaluation of the state tying. We plan to train the multi-mixture models from single-mixture models generated by the state tying, then apply these models into ISIP decoder, and evaluate the performance in terms of speed and accuracy.

Secondly, we plan to generalize the tied states. Currently the way we constructed the decision trees is only locally optimal. It is expected that the decision trees can be optimized by allowing both splitting and merging [1]. So in the next step, we are going to add a merging pass into the decision tree state tying module and improve the performance of the models further.

7. REFERENCES

[1] J.J. Odell, “The Use of Context in Large Vocabulary Speech Recognition,” Ph.D. Thesis, Cambridge University, 1995.

[2] R. Cordoba and J. Pardo, “Different Strategies for Distribution Clustering Using Discrete, Semicontinuous and Continuous HMMs in CSR,” *Proceedings of the Fourth International Conference on Spoken Language Processing*, pp. 1101-1104, Philadelphia, Pennsylvania, USA, October 1996.

[3] B. Gilles and K. Patrick, “Optimal Tying of HMM Mixture Densities Using Decision Trees,” *Proceedings of the Fourth International Conference on Spoken Language Processing*, pp. 350-353, Philadelphia, Pennsylvania, USA, October 1996.

[4] L.E. Baum, “An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes,” *Inequalities*, Vol. 1, pp. 1-8,

1972.

[5] V.V. Digalakis, M. Ostendorf and J.R. Rohlicek, “Fast Algorithms for Phone Classification and Recognition Using Segment-Based Models,” *IEEE Transactions on Signal Processing*, Vol. 40, pp. 2885-2896, 1992.

[6] X. Aubert and P. Beyerlein, “A Bottom-Up Approach for Handling Unseen Triphones in Large Vocabulary Continuers Speech Recognition,” *Proceedings of the Fourth International Conference on Spoken Language Processing*, pp. 14-17, Philadelphia, Pennsylvania, USA, October 1996.

[7] L. Deng and J. Wu, “Hierarchical Partition of the Articulatory State Space for Overlapping-feature Based Speech Recognition,” *Proceedings of the Fourth International Conference on Spoken Language Processing*, pp. 2266-2269, Philadelphia, Pennsylvania, USA, October 1996.

[8] L. Ariane, N. Yves and K. Roland, “Improving Decision Trees for Acoustic Modeling,” *Proceedings of the Fourth International Conference on Spoken Language Processing*, pp. 1053-1056, Philadelphia, Pennsylvania, USA, October 1996.

[9] W. Reichl and W. Chou, “Decision Tree State Tying Based Segmental Clustering for Acoustic Modeling,” *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 801-804, Seattle, Washington, USA, May 1998.

[10] K. Beulen and H. Ney, “Automatic Question Generation for Decision Tree Based State Tying,” *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 805-808, Seattle, Washington, USA, May 1998.

[11] *http://www.*isip.msstate.edu*/projects/speech/*, Institute for Signal and Information Processing, Mississippi State University.

APPENDIX A. UTILITIES OF STATE TYING

```
name: tie_state
synopsis: tie_state -p params.text

description:

this is the isip tie_state utility. All parameters are defined in the parameter file.


# parameters file for state tying using decision tree
#

# define the mode
#
# mode = train
mode = test

# common parameters for both mode
#
monophone_file = monophones.text
membership_file = isip_questions.text
out_models_file = new_models_test.text

input_mode = binary
output_mode = ascii
debug_level = 0

# special parameters for train mode
#
out_tree_file = data/trees.text
out_states_file = data/new_states.text

in_states_file = states.bin
in_models_file = new_models_train.text
special_models_file = spe_models.text
occupancy_file = occupancy_full.list

split_threshold = 15000.0
merge_threshold = -0.5
occupancy_threshold = 100
num_context = 1

# special parameters for test mode
#
in_tree_file = trees.text
in_phlist_file = all_wint_triphones.list
out_clist_file = clist.text

#
# end of file
```

APPENDIX B. DERIVATION

This appendix is to derive $\sum_t \left[(x_t - \mu)^T \cdot C^{-1} \cdot (x_t - \mu) \cdot \sum_s \Upsilon_s(t) \right] = n \cdot \sum_t \sum_s \Upsilon_s(t)$, which is missing from the derivation of the likelihood computation. In the following derivation, for the sake of simplicity, we use $\Upsilon(t)$ to denote $\sum_s \Upsilon_s(t)$.

The covariance matrix

$$C = \frac{\sum_t [\Upsilon(t) \cdot (x_t - \mu) \cdot (x_t - \mu)^T]}{\sum_t \Upsilon(t)} \tag{1}$$

Since the sum of occupancies $\sum_t \Upsilon(t)$ is a scalar, it can be moved to the left hand side.

$$\left(\sum_t \Upsilon(t) \right) \cdot C = \sum_t [\Upsilon(t) \cdot (x_t - \mu) \cdot (x_t - \mu)^T] \tag{2}$$

C is an $n \times n$ matrix. Right multiply C^{-1} on both sides of equation (2), and we can get

$$\left(\sum_t \Upsilon(t) \right) \cdot I = \left\{ \sum_t [\Upsilon(t) \cdot (x_t - \mu) \cdot (x_t - \mu)^T] \right\} \cdot C^{-1} \tag{3}$$

$$\overset{a}{=} \sum_t [\Upsilon(t) \cdot (x_t - \mu) \cdot (x_t - \mu)^T \cdot C^{-1}]$$

$$\overset{b}{=} \sum_t \left\{ \Upsilon(t) \cdot (x_t - \mu) \cdot [(x_t - \mu)^T \cdot C^{-1}] \right\}$$

where I is an identity matrix of order n, n is the order of the Gaussian Distribution. ‘a’ follows the distributive property of matrix, $(X + Y) \cdot Z = X \cdot Z + Y \cdot Z$, so C^{-1} can be moved into the Σ; ‘b’ follows the associative property of matrix, $(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$.

Define $B_t = (x_t - \mu)$, which is a column vector. $(x_t - \mu)^T$ is a row vector, C^{-1} is an $n \times n$ matrix, so $(x_t - \mu)^T \cdot C^{-1}$ is also a row vector. Define $A_t = (x_t - \mu)^T \cdot C^{-1}$. Now, equation (3)

can be simplified as

$$\sum_t \{\Upsilon(t) \cdot B_t \cdot A_t\} = \left(\sum_t \Upsilon(t)\right) \cdot I \tag{4}$$

What we want to get for computing the likelihood is

$$\sum_t [(x_t - \mu)^T \cdot C^{-1} \cdot (x_t - \mu) \cdot \Upsilon(t)] = \sum_t \{\Upsilon(t) \cdot A_t \cdot B_t\} \tag{5}$$

Now we know $\sum_t \{\Upsilon(t) \cdot B_t \cdot A_t\} = \left(\sum_t \Upsilon(t)\right) \cdot I$ from equation (4), what we need to do is to get the value of $\sum_t \{\Upsilon(t) \cdot A_t \cdot B_t\}$ using this information.

For multiplication of any row vector A and any column vector B, there is

$$B \cdot A = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix} \cdot \begin{bmatrix} a_1 & a_2 & \dots, a_n \end{bmatrix} = \begin{bmatrix} b_1a_1 & b_1a_2 & \dots & b_1a_n \\ b_2a_1 & b_2a_2 & \dots & b_2a_n \\ \dots & \dots & \dots & \dots \\ b_na_1 & b_na_2 & \dots & b_na_n \end{bmatrix} \tag{6}$$

$$A \cdot B = \begin{bmatrix} a_1 & a_2 & \dots, a_n \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix} = a_1b_1 + a_2b_2 + \dots + a_nb_n \tag{7}$$

So $A \cdot B = trace(B \cdot A)$ (8)

For $\sum_i \{B_i \cdot A_i\}$ and $\sum_i \{A_i \cdot B_i\}$, where each A_i is a row vector, and each B_i is a column vector,

$$\sum_i \{A_i \cdot B_i\} = \sum_i trace\{B_i \cdot A_i\} = trace\left(\sum_i \{B_i \cdot A_i\}\right) \tag{9}$$

this is a scalar.

So now, knowing $\sum_t \{\Upsilon(t) \cdot B_t \cdot A_t\} = \left(\sum_t \Upsilon(t)\right) \cdot I$, it's easy to get

$$\sum_t \{\Upsilon(t) \cdot A_t \cdot B_t\} = trace\left(\sum_t \{\Upsilon(t) \cdot B_t \cdot A_t\}\right) \quad (10)$$
$$= trace\left(\left(\sum_t \Upsilon(t)\right) \cdot I\right) = n \cdot \sum_t \Upsilon(t)$$

The n on the right hand side is the trace of the identity matrix I of order n. Replace what we have defined $A_t = (x_t - \mu)^T \cdot C^{-1}$, $B_t = (x_t - \mu)$, we get

$$\sum_t (x_t - \mu)^T \cdot C^{-1} \cdot (x_t - \mu) \cdot \Upsilon(t) = n \cdot \sum_t \Upsilon(t) \quad (11)$$

$$\sum_t \left[(x_t - \mu)^T \cdot C^{-1} \cdot (x_t - \mu) \cdot \sum_s \Upsilon_s(t)\right] = n \cdot \sum_t \sum_s \Upsilon_s(t) \quad (12)$$