IEICE TRANS. INF. & SYST., VOL.E94–D, NO.11 NOVEMBER 2011

PAPER

Decision Tree-Based Acoustic Models for Speech Recognition with Improved Smoothness

Masami AKAMINE[†a)], *Member and* **Jitendra AJMERA**[††], *Nonmember*

SUMMARY This paper proposes likelihood smoothing techniques to improve decision tree-based acoustic models, where decision trees are used as replacements for Gaussian mixture models to compute the observation likelihoods for a given HMM state in a speech recognition system. Decision trees have a number of advantageous properties, such as not imposing restrictions on the number or types of features, and automatically performing feature selection. This paper describes basic configurations of decision tree-based acoustic models and proposes two methods to improve the robustness of the basic model: DT mixture models and soft decisions for continuous features. Experimental results for the Aurora 2 speech database show that a system using decision trees offers state-of-the-art performance, even without taking advantage of its full potential and soft decisions improve the performance of DT-based acoustic models with 16.8% relative error rate reduction over hard decisions.

key words: *speech recognition, acoustic modeling, decision trees, probability estimation, likelihood computation*

1. Introduction

The acoustic model in automatic speech recognition (ASR) based on hidden Markov models (HMMs) provides the decoder with the likelihood of an observation given a specific HMM state. State-of-the-art speech recognizers commonly use Gaussian mixture models (GMMs) or artificial neural networks (ANNs) to compute the desired likelihoods [1].

While decision trees (DTs) are powerful statistical tools and have been widely used for many pattern recognition applications, their effective usage in ASR is limited to state tying prior to building context-dependent acoustic models [2]. Recently, some work has been reported in the literature where DTs were used to perform the likelihood computation [3], [4]. DTs determine the likelihood by asking a series of questions about the observation. Starting from the root node of the tree, appropriate questions are asked at each level. Based on the answer to the question, an appropriate child node is selected and evaluated next. This process is repeated until the selected node is a leaf node, which provides the pre-computed likelihood of the observation given the DT model.

In this paper, we propose new methods to use DTs as acoustic models instead of GMMs. The work here is closely related to that in [3], but our approach and focus are different. In [3], Foote exploited DTs for improving vector quantization in discrete acoustic models and gave a training method to binary trees with hard decisions. We view a decision tree as a tree-based model with an integrated decision-making component. Decision tree-based acoustic models have a number of drawbacks, and these have not been considered sufficiently [3], [4]. One of the most serious problems is vulnerability to noise or any mismatch in feature statistics between training and recognition. A small change in a feature value might result in a large change in the likelihood. The likelihood given by DTs is not smooth.

The motivation for using DTs as acoustic models comes from a number of interesting properties of DTs that might prove to be advantageous for speech recognition. For instance:

- DTs do not impose any restrictions on the features or on their distributions.
- Information from many different sources, ranging from low-level acoustics to high-level grammar-related sources, can be integrated efficiently.
- Automatic state-tying can be incorporated.
- Owing to the conditional nature of the evaluation process of a DT, the subset of features actually used during recognition depends on the input signal.

Our ultimate goal is to take advantage of the decision-making aspect of decision tree acoustic models by providing the model with high-level information capable of effectively specializing the model for particular conditions. For example, if a DT asks a question about the gender of the speaker, then the child branches are in effect gender-dependent models. The question at a node can involve a scalar or a vector value. Droppo et al. explored decision trees with vector-valued questions [4]. We have dealt with scalar-valued questions [5] because we want to explore the advantages of the above-mentioned DT-based acoustic model. It is difficult to deal with categorical questions and the mix of categorical and numerical questions in vector valued questions.

In this paper, we propose smoothing techniques for decision tree-based acoustic models to improve the robustness against noise. We describe basic configurations of DTs as well as training of DT-based acoustic models. Then we propose two techniques to improve the robustness of the basic model: DT mixture models and soft decisions for continuous features. In the proposed methods, soft-decision trees are created using two different methods: one that converts hard decisions to soft ones and the other that trains soft-decision trees from scratch.

Manuscript received February 14, 2011.
Manuscript revised June 28, 2011.
[†]The author is with Corporate Research & Development Center, Toshiba Corp., Kawasaki-shi, 212–8582 Japan.
[††]The author is with IBM Research Lab., New Delhi, India.
a) E-mail: masa.akamine@toshiba.co.jp
DOI: 10.1587/transinf.E94.D.2250

The remainder of this paper is organized as follows. Section 2 describes DT acoustic models and explains how the models are trained and evaluated. Section 3 proposes techniques to improve the robustness of DT acoustic models. Section 4 presents the experimental setup and results obtained using the Aurora 2 speech database. Finally, Sect. 5 concludes this paper.

2. Decision Tree-Based Acoustic Models

2.1 Basic Configuration

As shown in Fig. 1, DT acoustic models are HMM-based acoustic models that utilize decision trees instead of, for instance, GMMs or ANNs to compute observation likelihoods. A decision tree determines the likelihood of an observation by asking a series of questions about the observation. The decision tree is discriminative when it is used as a classifier. However, we use decision trees in HMM acoustic models instead of GMM to compute the likelihood. Therefore, decision tree-based acoustic models are generative. Questions are asked at question nodes, starting at the root node of the tree. Based on the result of the question, the appropriate child node is selected and evaluated next. This process is repeated until the selected node is a leaf node, which contains the pre-computed likelihood of the observation given the model. The likelihood value is computed and stored in each leaf during the training process.

Throughout this paper, we assume that decision trees are implemented as binary trees. DTs can deal with multiple target classes simultaneously [6] and this makes it possible to use a single DT for all HMM states. However, we found from preliminary experiments that better results are obtained by using a different tree for each HMM state.

In this configuration, each DT is trained to maximize likelihoods for the training data that corresponds to the associated HMM state. The training samples corresponding to the state are referred to as "true" samples and the other samples are referred to as "false" samples. Similar to training conventional HMM acoustic models, discriminative training is possible for the DT-based acoustic models. However, we focus on maximum likelihood (ML) training in this paper. Discriminative training is a subject for future work. The scaled likelihood of the D dimensional observation $X = (x_1, x_2, x_d, \ldots, x_D)$ given state q can then be computed using:

$$L(X|q) = \frac{P(q|X)}{P(q)} \tag{1}$$

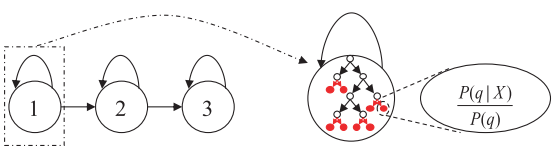

Fig. 1 Decision tree-based acoustic models. Decision trees are used to compute observation likelihoods for every state of the HMM.

where $P(q|X)$ is the posterior probability of state q given observation X and $P(q)$ is the prior probability of state q. The likelihood given by the above equation is stored in each leaf node.

2.2 Training of Decision Tree-Based Acoustic Models

The high-level training algorithm of DT acoustic models is similar to that of standard GMM systems. It consists of an initialization stage, followed by one or more stages that change the complexity and/or structure of the model. The initial model is created by using state-level forced alignments from an existing system, or by using a flat start approach [7]. The individual DT models are constructed using an algorithm similar to the standard C4.5 algorithm [8] for training decision trees. It consists of a growing stage and a bottom-up pruning stage. A DT is grown by turning a leaf node into a question node and producing two new child leaf nodes. This is also referred to as splitting. The training algorithm evaluates all possible splits of a node, i.e. evaluating every feature and corresponding threshold, and selects the split that maximizes the split criterion and meets a number of other requirements. Specifically, splits must pass a chi-square test and must result in leaves with a sufficiently large number of samples. The split criterion is the total log likelihood increase of the true samples. In other words, trees are grown to maximize the total (scaled) log likelihood of the true training samples. If the number of true samples reaching a node (node d) is N_T and the total number of samples (true and false) is N_{all}, the increase of the total log likelihood ΔL from the splitting is described by

$$\Delta L = N_T^y \log L_d^y + N_T^n \log L_d^n - N_T \log L_d \tag{2}$$

$$L_d = \frac{N_T}{N_{all}} \cdot \frac{1}{P(q)}, \quad L_d^y = \frac{N_T^y}{N_{all}^y} \cdot \frac{1}{P(q)},$$

$$L_d^n = \frac{N_T^n}{N_{all}^n} \cdot \frac{1}{P(q)} \tag{3}$$

where L_d, L_d^y, and L_d^n are the likelihood at node d, at a child node answering the split question with *yes* (denoted "child *yes*"), and at the other child node answering with *no* (denoted "child *no*"), respectively. N_T^y and N_{all}^y are the number of the true and all samples at child *yes*, N_T^n and N_{all}^n are the number of the true and all samples at child *no*, respectively, as shown in Fig. 2. $P(q)$ is the prior probability of state q.

Once a tree is fully grown, the likelihood split criterion

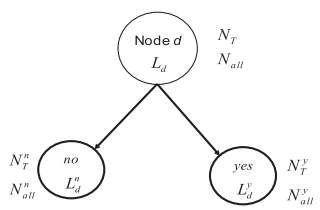

Fig. 2 A split based on a question that maximizes log likelihood increase.

is used to prune the tree in a bottom-up, worst-first fashion to give the desired model complexity. Finally, Laplace (or add-one) smoothing is used to increase the robustness of the counts used to estimate the leaf values (i.e., the likelihoods) [6].

After initial DTs are constructed from the training alignments, the HMM transition parameters and DT leaf values are re-estimated using several iterations of the Baum-Welch algorithm [1]. Depending on the quality of the initial alignments, the process of growing trees and re-estimating the parameters can be repeated until a desired stopping criterion has been reached, such as a maximum number of iterations.

3. Smoothness Improvement

One undesirable property of the DT acoustic models described above is that the likelihood as a function of the features is not smooth; a minute change in a feature value might result in a large change in the likelihood. By using large amounts of training data, robust features and big trees, this problem can be alleviated, but in practice, the situation is often less than ideal. We propose two techniques to remedy this issue.

3.1 Mixture Models

A popular and very effective technique for improving the robustness and accuracy of DTs is to use an ensemble method, such as boosting [9] or bagging [10]. The resulting ensembles are also referred to as forests. A forest is evaluated by combining (e.g., averaging) the results of all individual trees. A small change in a feature value might still cause a large change in the output of one tree, but generally not in all trees at once. Consequently, the output of a forest is much smoother than the output of any individual tree.

However, there is a downside to a method such as bagging. Instead of a single tree, a bagging approach uses multiple trees to compute the likelihood as mentioned above. Each tree is constructed to cover the entire acoustic space and its size is the same as the original single tree. Therefore, the bagging approach results in a greater number of parameters in general. Although it is possible to apply a certain restriction such as using a much smaller number of parameters for each tree to control the total number of parameters in the bagging, this would harm the performance.

To overcome this problem, we propose decision tree mixture models, where each mixture component models a different part of the acoustic space. The likelihood of the mixture corresponds to the weighted sum of the likelihoods of the individual components as follows:

$$L(X|q) = \sum_{k=1}^{N} w_k L(X|M^k) \tag{4}$$

where N is the number of components, w_k is the weight of component k, and M^k corresponds to the DT model of component k for state q. Note that it follows from this formulation that each component is trained to discriminate between data that corresponds to the component itself and all other data, including data from the other components in the same mixture. Mixture models benefit from the smoothing property of ensemble methods and since each tree is responsible for only a portion of the acoustic space, the number of components and the complexity of each component can be traded off.

Initialization of a DT mixture model is non-trivial and for our experiments, we used an *ad hoc* method that randomly assigns true data samples to each component. Once the initial components are constructed, the EM algorithm is used to update the mixture weights and leaf values of the components. The resulting mixture model is subsequently used to create new component training data sets, which are employed to re-train the components from scratch. This process is repeated for a certain number of iterations. The algorithm works as follows:

Create N partitions of the data D_q belonging to State q. Denote each partitioned data as $D_1 \cdots D_N$.

For each k in 1 ... N, do:
  Train the initial model M^k using D_k as "true" samples and all other data as "false" samples.
end
For iterations in 1...I, do:
  For each data point X in D_q do:
    Compute the likelihoods of X given the DT models $M^1 \cdots M^N$.
    Assign X to the DT model that provides maximum likelihood as "true" samples.
  end
  For each k in 1...N do:
    Train DT model M^k using samples assigned as "true" samples, and all other samples as "false" samples.
  end
end

3.2 Soft-Decision Trees

Another method that can be used to improve the robustness of DTs is to make "soft" or "fuzzy" decisions, instead of hard decisions [11]. The main idea is that in a region around the decision threshold the weighted contribution of both child branches is used. The weighting is a function of the distance between the feature value and the decision threshold.

Our approach is to treat a soft-decision result as a probability, the probability that the observation corresponds to one child branch versus the other child branch. We model this probability by using a sigmoid function given by

$$f_d(x) = \frac{1}{1 + \exp(s_d(t_d - x))} \tag{5}$$

where t_d is the threshold of question at node d and s_d is a parameter that determines the smoothness of the decision.

If this parameter is set to infinity, the decision reduces to a hard, binary decision. The total likelihood of observation X for given state q returned by the question at node d is given by

$$L_d(X|q) = f_d(x_d)L_d^y(X|q) + (1 - f_d(x_d))L_d^n(X|q) \quad (6)$$

where x_d is the feature value that node d asks about, $L_d^y(X|q)$ is the likelihood that the child answering the question at node d with *yes* ("child *yes*") returns and $L_d^n(X|q)$ is the likelihood that child *no* returns.

We explored two ways of training the parameters of soft-decision trees. The following subsections explain these two methods in detail.

3.2.1 Convert Hard to Soft-Decision Trees

Since the likelihood of any given node depends on the likelihoods of its children nodes for any observation, the estimation of maximum likelihood (ML) parameters of node d, (t_d and s_d) also requires likelihoods of children nodes. The children likelihoods, in turn, require likelihoods further down the tree. This means that for ML estimation of question parameters of a soft DT, one has to traverse the tree completely.

One way of tackling this problem is to convert hard decisions to soft ones once hard decision trees are trained. Keeping the tree structure unchanged, we re-estimate the threshold parameter t_d and estimate the smoothness parameter s_d using an iterative gradient-based optimization algorithm that tries to maximize the total training data likelihood. Our implementation is based on the RProp algorithm [12] and automatically adapts the learning rate and batch size for each parameter individually. All question parameters are re-estimated simultaneously. After every iteration, the leaf values of the trees are re-estimated based on the new parameter values t_d and s_d. The optimization steps are as follows:

For i=1…I iterations do:
  For b=1..B_i batches do:
    For n=1..N tree nodes do:
      Compute the derivative of the total log likelihood with regard to smoothness and threshold parameters, respectively
      Update parameter by stepping in the direction of the gradient (to maximize the total log likelihood). Step size (learning rate) is determined by RProp.
  Update learning rate and, if needed, batch size B_i.
Update all leaf parameters (assign data points to leaves according to posterior probability of leaf given data point).

The derivatives of the log likelihood of Eq. (6) with regard to the smoothness parameter s_d and the threshold parameter t_d are given by

$$\frac{\partial \log(L_d(X|q))}{\partial s_d} = \frac{1}{L_d(X|q)} \frac{\partial L_d(X|q)}{\partial s_d}$$

$$= \left\{ \frac{L_d^y(X|q) - L_d^n(X|q)}{L_d(X|q)} \right\} \frac{\partial f_d(x_d)}{\partial s_d} \quad (7)$$

$$\frac{\partial \log(L_d(X|q))}{\partial t_d}$$

$$= \left\{ \frac{L_d^y(X|q) - L_d^n(X|q)}{L_d(X|q)} \right\} \frac{\partial f_d(x_d)}{\partial t_d} \quad (8)$$

where

$$\frac{\partial f_d(x_d)}{\partial s_d} = \frac{(t_d - x_d)\exp(s_d(t_d - x_d))}{(1 + \exp(s_d(t_d - x_d)))^2} \quad (9)$$

$$\frac{\partial f_d(x_d)}{\partial t_d} = \frac{s_d \cdot \exp(s_d(t_d - x_d))}{(1 + \exp(s_d(t_d - x_d)))^2} \quad (10)$$

Features that already represent a soft decision can be incorporated directly into a DT by using pass-through questions. A pass-through question simply returns the feature value itself as the soft answer. An example of such a feature is a gender-classification feature that represents the probability that the speaker is female.

3.2.2 Train Soft-Decision Trees From Scratch

One problem with the approach in Sect. 3.2.1 is that once a hard decision tree is trained, the features to be evaluated at different nodes are fixed and only the smoothness and threshold parameters are re-estimated. It can be argued though that a soft question based on another feature may better split data at a particular node. Therefore, a soft DT trained from scratch should be different from a soft DT converted from a hard DT. This section shows how we can train the soft DTs from scratch.

Similar to the training of hard DTs, all possible soft-splits are evaluated at every node while growing the tree. This means finding the best feature x, smoothness parameter s_d and threshold t_d at node d. The soft-question that results in maximum log-likelihood increase of true samples is selected for that node. As mentioned earlier, this requires likelihood values of the two children nodes.

These likelihood values are estimated by accumulating answers to the soft question for every training data point x_i as follows. First, posterior probabilities for true samples at child nodes are given by

$$P(yes|X \in D_T) = \frac{\sum_{i=1}^{N_T} \frac{f_d(x_i)L_d^y(X|q)}{L_d(X|q)}}{\sum_{i=1}^{N_{all}} \frac{f_d(x_i)L_d^y(X|q)}{L_d(X|q)}} \quad (11)$$

$$P(no|X \in D_T) = \frac{\sum_{i=1}^{N_T} \frac{(1-f_d(x_i))L_d^n(X|q)}{L_d(X|q)}}{\sum_{i=1}^{N_{all}} \frac{(1-f_d(x_i))L_d^n(X|q)}{L_d(X|q)}} \quad (12)$$

where $f_d(x_i)$ is computed using Eq. (5). N_T and N_{all} denote the number of true and all samples at node d, respectively. Posterior probabilities for negative samples are complementary to those of positive samples, i.e. $P(\cdot|X \in D_F) = 1 - P(\cdot|X \in D_T)$.

The posterior probabilities are converted to scaled likelihoods by Bayes rule as follows:

$$L_d^y(X|q) = \frac{P(yes|X \in D_T)}{P(q)} \tag{13}$$

$$L_d^n(X|q) = \frac{P(no|X \in D_T)}{P(q)} \tag{14}$$

where $P(q)$ is the prior probability of the given state.

Equations (11)–(14) show that the likelihood computation for children nodes is an iterative process and depends on soft-question parameters of the parent node. These likelihoods can then be used for updating smoothness parameter s_d and threshold parameter t_d using RProp algorithm [12]. Let t denote the current iteration (epoch). The smoothness parameter s_d and the threshold parameter t_d are changed according to:

$$s_d(t+1) = s_d(t) - \text{sign}\left(\frac{\partial \log L_d(X|q)}{\partial s_d}\right) \cdot \delta_t \tag{15}$$

$$t_d(t+1) = t_d(t) - \text{sign}\left(\frac{\partial \log L_d(X|q)}{\partial t_d}\right) \cdot \delta_t \tag{16}$$

where the partial derivatives of the log likelihood at node d with regard to smoothness and threshold parameters are given by Eq. (7) and Eq. (8).

The individual step size δ_t is adapted based on the change of sign of the derivative of the log likelihood with regard to the corresponding parameter. The likelihoods of children nodes $L_d^y(X|q)$ and $L_d^n(X|q)$ in Eq. (7) and (8) are computed from Eq. (13) and Eq. (14). The initial value of threshold parameter t_d corresponds to the best hard split that maximizes the total log likelihood of true data, and initial value of smoothness parameter s_d is decided based on the standard deviation of feature values $x_i; i = 1, \ldots, N_{all}$. The above parameter updating and likelihood computation processes are repeated for a number of iterations. The algorithm works as follows:

At each node,
For $i = 1 \ldots I$ iterations do:
For each data point X in D_q do:
Compute posterior probabilities $P(yes|X \in D_T)$ and $P(no|X \in D_T)$ using Eq. (11) and (12), for true samples and complementary values for false samples, respectively.
Compute likelihoods $L_d^y(X|q)$ and $L_d^n(X|q)$ using Eq. (13) and (14), respectively.
Compute smoothness parameter $s_d(t)$ and threshold parameter $t_d(t)$ using Eq. (15) and (16), respectively.
end
end
If the total likelihood computed by Eq. (6) is bigger than that at the upper node with a predetermined margin, split the node with the selected question.
Else, end.

Once a soft question for node d is selected, the training data is split probabilistically among *yes* and *no* children nodes. The split probability is given by Eq. (5). The children nodes are then grown similarly in a depth-first fashion.

This process continues until:

- *The in-class and other-class separation of training data at a node has minimum counts of both categories.*
- *This separation also follows the chi-square statistical significance test.*

It is possible that sometimes there is no good soft-question for a node but the corresponding hard-question satisfies the above-mentioned requirements. In this situation, a hard question $x \leq t_d$ is used for that node. Therefore the resulting tree has a mix of soft and hard questions.

4. Experiments and Results

All experimental results are based on the Aurora 2 speech database [13]. The task is connected digit string recognition under noisy conditions for the American-English language. This speech database was selected, because the task is a pure acoustic modeling problem, the baseline error rates are relatively high and the database is small enough to be able to quickly run experiments. The downside of using the Aurora 2 database is that some of the more interesting aspects of DT acoustic models, such as integrated state-tying and incorporation of grammar-level features, cannot be explored properly.

For all experiments, we used the multi-condition training data set, which contains a total of 8440 utterances at 5 different SNR levels (5 dB - clean). There are 3 test sets: set A contains noises seen in the training data, set B contains unseen noises, and set C contains convolutional distortions applied to one noise condition from both set A and B. The test data set contains data from 7 different SNR levels (−5 dB - clean), but only results for SNR levels between 0 dB and 20 dB are used to compute the average word accuracy numbers.

The HMM structure and model complexity is identical for all systems. The model complexity of a DT is taken to be equal to the total number of nodes in the DT, since each question refers to a single threshold and each leaf stores a single likelihood. There are 11 digit models (oh, zero, one, ..., nine) and 2 silence models (sil and sp). The digit models are each composed of 16 states; sil has 3 states; sp consists of a single state. The GMM baseline system uses 3 diagonal Gaussians per GMM (6 Gaussians per GMM for silence).

4.1 DT Acoustic Models

Table 1 shows the average word accuracy of a standard GMM system and a comparable DT system. Results for two different feature sets are shown. One feature set consists of 12 static PLP features and log energy, together with their 1st and 2nd derivatives, totaling 39 features. The static features are normalized using cepstral mean normalization and the static log energy feature is normalized with respect to the maximum value in the utterance. This feature set is denoted by "PLP + E". The other feature set is similar, but instead of 3 log energy-based features, it contains 8 log filter-bank features plus their 1st, 2nd and 3rd derivatives. The total

Table 1 Average word accuracy of conventional GMM systems and proposed DT systems for two different feature sets.

System (feature), Number of parameters	Set A	Set B	Set C	Ave.
GMM 1 (PLP+E), 44k	**89.2**	**88.6**	**88.7**	**88.8**
GMM 2 (PLP+FB), 88k	**86.5**	**86.5**	**86.5**	**86.5**
DT 1 (PLP+E), 44k	**85.1**	**81.7**	**81.0**	**82.6**
DT 2 (PLP+FB), 44k	**89.1**	**87.8**	**89.1**	**88.7**

number of features is 68 in this case. The decision to use the filter-bank features instead of the energy features was taken based on an analysis of the performance of PLP+E features for DT-based acoustic models. The analysis is given later in this section. The filter banks cover the whole frequency range and are equally spaced in the mel scale.

The DT systems use a single tree per HMM state and the model complexity of each tree is the same as that of the corresponding GMM in the baseline system. This corresponds to 118 leaf nodes per tree. The DTs use hard decisions.

The results show that the word accuracy of DT system 2 is very similar to that of GMM system 1. In fact, the GMM system is better on only 27 of 50 individual test conditions. Although the feature set used by DT system 2 is larger than the feature set used by GMM system 1, the number of model parameters is equivalent. That is one of the advantages of DT acoustic models: the number of model parameters is completely independent of the size of the feature set.

We can see from the results that in terms of the feature set, what works for a DT system does not necessarily work for a GMM system, and vice versa. That is not surprising, since the models are fundamentally different. For example, the GMM system uses diagonal Gaussians, and hence it assumes that the features are uncorrelated, unlike DTs. However, that assumption does not hold for feature set 2, which might explain the performance degradation of GMM system 2 compared to GMM system 1. Another example is that GMMs always use all features for the likelihood computation, whereas DTs use a variable number of features. Owing to the relatively small model complexity, some trees contain leaf nodes that are only 2 questions removed from the root of the tree and the average tree depth is merely 8.9 questions deep. The consequence is that even a small number of corrupted features can greatly affect the accuracy of the model. This is the reason for the large performance gap between DT systems 1 and 2. By keeping track of the number of times each feature is evaluated during recognition, we found that the 3 energy features (i.e., the static feature and the two derivatives) together account for more than 40% of the total number of feature evaluations for DT system 1. Given that energy is useful for discriminating speech from silence and that each tree is trained using all training data samples, 33% of which corresponds to silence, it can be explained why energy features are used so frequently. Unfortunately, energy is not a particularly robust feature, since it is calculated over the whole spectrum. Replacing the energy-based features with filter-bank features alleviates this problem and gives a significant boost in accuracy.

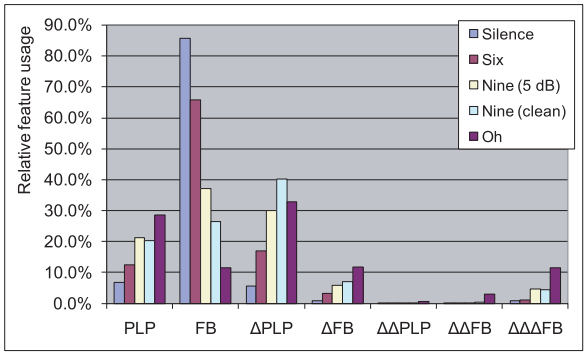

Fig. 3 Relative feature usage for different models and noise conditions in the proposed DT system DT2.

Table 2 Comparison of the word accuracy of a single DT system and a 3-DT-mixture system.

System (feature), mixture of trees	Set A	Set B	Set C	Ave.
DT 2 (PLP+FB), single	**89.1**	**87.8**	**89.1**	**88.7**
DT 3 (PLP+FB), 3 mixtures	**89.5**	**88.5**	**89.6**	**89.2**
DT bagging (PLP+FB)	**88.6**	**79.2**	**82.1**	**83.3**

Figure 3 shows the relative usage of features for models "oh (clean)", "six (clean)", "nine at 5 dB", "nine in clean" and "silence". The usage was counted for PLP and FB features, and their dynamic features. From this figure, we can see that the features used in decoding are selected automatically in the DT-based acoustic models and that feature usage can be easily analyzed in DT-based systems.

4.2 Mixture Models

In the following experiment, mixtures of trees are used to model each HMM state. The mixture models have 3 components (trees) per state. The number of parameters in each component is fixed and it corresponds to 39 leaves per tree (78 for silence).

Table 2 shows the results. For the purpose of comparison, a DT bagging system is also shown. The bagging system has 25 trees each state as the ensemble. The number of trees was experimentally determined as a good number. Each tree was trained using the data sampled randomly from the training data. The results from 25 trees were combined by averaging. All models in Table 2 have the same number of parameters. The results show that the bagging system is inferior to the others. Its performance becomes worse than that of the DT single model, especially for Set B and Set C. The bad performance for Set B and Set C can be attributed to over-fitting to the training data. The DT mixture model outperforms the DT baseline system on 46 of 50 individual test conditions. Comparing the performance with the GMM system shows that the DT mixture model is a competitive

model. In fact, the DT mixture system scores worse on only 14 of 50 individual test conditions.

The baseline DT system can generate at most 118 unique likelihoods per model, since that is the number of leaves per tree. On the other hand, the DT mixture system can generate no more than 39 unique likelihoods per tree, but since the contributions of 3 trees are added together, the resulting number of unique likelihoods can be much higher. We found that on the training data each mixture generates roughly 2700 unique likelihoods on average. This demonstrates that without changing the total number of parameters, mixture models can provide a dramatic increase in resolution, resulting in models that are more robust.

4.3 Soft Decision Trees

We ran some preliminary experiments to verify the advantage of soft DTs trained from scratch over soft DTs converted from hard DTs. This was done using a subset of 400 training utterances and 1500 test utterances. Table 3 shows speech recognition performance and the number of parameters for these two soft DT systems. For the soft DT system converted from hard DTs, we took the hard decision model ("DT 2, single") and then converted it to a soft decision model according to the method described in Sect. 3.2.1. For each question node d, the initial estimate of the smoothness parameter was based on the variance of the feature at node d. After initialization of the smoothness parameters, all model parameters were re-estimated by running the optimization algorithm for 3×20 iterations. Note that the structure of the trees does not change. We created the other soft DT system from scratch using the method in Sect. 3.2.2.

Results show that the soft DT system trained from scratch requires far fewer parameters to achieve a performance similar to that of the soft DT system converted from hard DTs. This means that, compared with the soft DTs converted from hard decisions, the soft DTs from scratch have better separation of in-class and other-class data points.

Since the soft DTs from scratch tend to have far fewer parameters, the bottom-up pruning is not necessarily required in many cases. However, the pruning can be done using development data apart from training data in the same way as for the hard DTs, i.e. worst-first fashion, in order to improve the robustness against unseen data.

For experiments with full training set, we compared two soft DT systems with the hard DT system. Table 4 presents average word accuracies for different test sets for the hard DT system (denoted "DT hard"), a soft DT system converted from hard DTs ("DT soft from hard") and a soft DT system trained from scratch ("DT soft from scratch").

Table 3 Comparison of the word accuracy and the number of parameters between soft DT systems converted from hard DTs and trained from scratch. PLP+FB was used as the feature set.

System	Set A	Set B	Set C	Ave.	Number of parameters
DT soft from hard	**87.4**	**85.0**	**87.5**	**86.6**	**50k**
DT soft from scratch	**86.3**	**86.0**	**88.4**	**86.9**	**33k**

Table 4 Comparison of the word accuracy of a hard DT system and soft DT systems.

System (feature)	Set A	Set B	Set C	Ave.
DT hard (PLP+FB)	**89.1**	**87.8**	**89.1**	**88.7**
DT soft from hard (PLP+FB)	**91.2**	**89.6**	**90.9**	**90.6**
DT soft from scratch (PLP+FB)	**90.9**	**89.6**	**90.6**	**90.4**

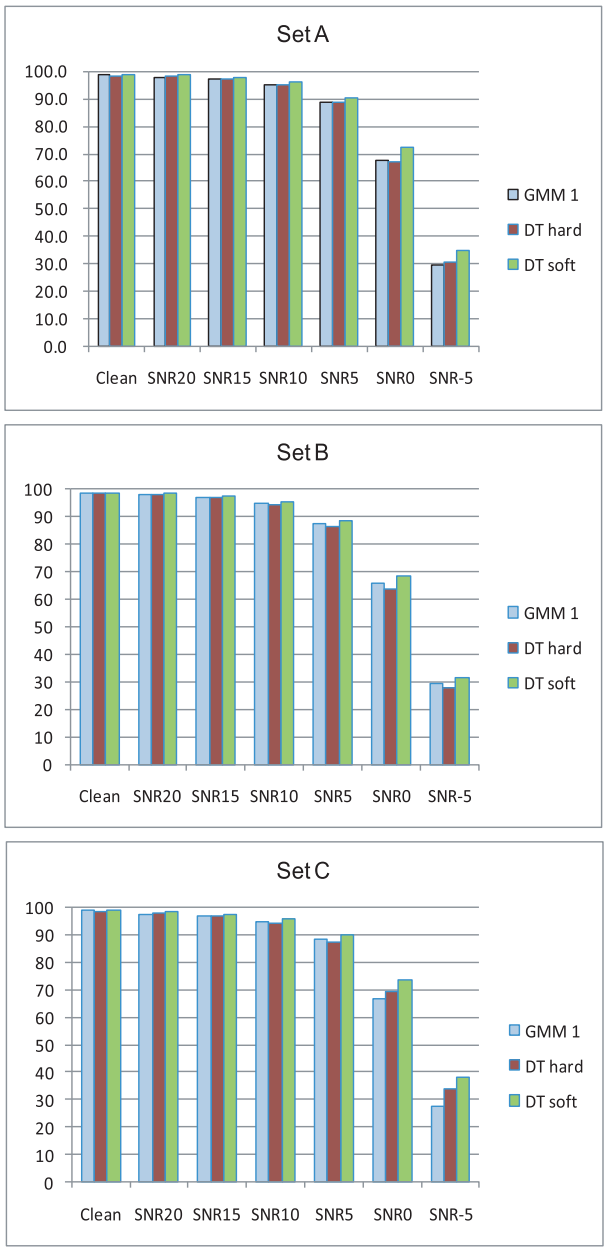

Fig. 4 Comparison of the word accuracy of GMM, hard DT and soft DT systems for different noise conditions.

We created these soft DT systems in the same way as described above. Note that the numbers of nodes in all DT systems in Table 4 are the same. The bottom-up pruning was applied to both "DT hard" and "DT hard to soft" systems to make the number of parameters the same. We stopped growing the tree for "DT soft from scratch" at the same number of parameters as the other systems.

Table 4 shows that the soft decision models are significantly better than the hard decision model. The soft DT system converted from hard DTs achieved 16.8% relative error rate reduction on average on three test sets over the hard DT system. Figure 4 shows the comparison between the GMM 1 system, hard and soft DT systems for clean and noise conditions for test sets A, B and C. It can be seen from Fig. 4 that the soft DTs improve the robustness against noise. The accuracy improvement of soft DTs from scratch was 15.0%, which is slightly smaller than that of converted soft DTs. We observed from the experiments that the soft DTs trained from scratch tend to be more robust in severer noise conditions. The performance of the soft DTs at -5 dB of SNR was 34.9% whereas that of soft DTs converted from hard DTs was 33.8%.

The experimental results show that soft DTs improve the robustness to noise as well as the performance in clean conditions. This is consistent with our intuition that soft decisions should be more capable of handling overlaps of features between different classes. However, soft DTs are computationally more expensive in training and testing than hard DTs. Hard DTs are fast in computing acoustic scores owing to the nature of binary decisions. There is a trade-off between computation and the performance.

5. Conclusions

This paper presented decision tree-based acoustic models for automatic speech recognition. We described the basic configuration of DT-based acoustic models and proposed DT mixture models and soft DTs to improve the smoothness of the acoustic models, which resulted in improved performance. Soft DTs are created by the proposed two methods: one method creates soft DTs by converting hard-questions to soft ones while maintaining the DT structure unchanged; the other method trains soft DTs from scratch.

Experimental results for the Aurora 2 speech database show that the performance of a system using decision trees offers state-of-the-art performance, even without taking advantage of its full potential. The performance of soft DT systems was shown to be significantly better than that of a hard DT system as well as better than that of a baseline GMM system. The soft DTs converted from hard DTs achieved 16.8% relative error rate reduction over the hard DT system.

We used conventional acoustic feature sets both for the proposed DT-based acoustic models and conventional GMMs for the purpose of comparison between these models. Future work includes investigating feature sets suitable for DT acoustic models. DTs can use not only low-level acoustic features but also high-level information such as questions about gender, phonetic contexts, acoustic environments and the speaker.

With that in mind, an interesting extension to the basic DT acoustic model configuration involves allowing DTs to perform automatic state tying, or more precisely, untying. We can extend the DT, allowing it to ask questions about features that can identify the HMM state for which the model is evaluated, and hence it can untie states. We call such features decoding features to distinguish them from acoustic features. The resulting tree can have an arbitrary mix of tied and untied parameters. This approach allows DTs to incorporate phonetic context-dependent state tying directly into the acoustic model. Furthermore, the amount of state tying can change depending on the input signal.

A decision tree is discriminative when it is seen as a classifier. However, the decision tree-based acoustic models described in this paper are generative because decision trees are used instead of GMM to compute the likelihood. While both ML and discriminative training are capable of training the proposed DT-based acoustic models, only ML training is considered in this paper. Discriminative training is one of the subjects for future work.

We want to explore these approaches in large vocabulary tasks.

References

[1] R. Cole, J. Mariani, H. Uszkoreit, G.B. Varile, A. Zaenen, A. Zampolli, and V. Zue (eds), Survey of the state of the Art in Human Language Technology, Cambridge University Press, New York, 1997.
[2] P.C. Woodland, J.J. Odell, V. Valtchev, and S.J. Young, "Large vocabulary continuous speech recognition using HTK," Proc. ICASSP 1994, pp.125–128, 1994.
[3] J.T. Foote, Decision-tree probability modeling for HMM speech recognition, Ph.D. thesis, Brown University, Providence, RI, 1993.
[4] J. Droppo, M.L. Seltzer, A. Acero, and Y.B. Chiu, "Towards a non-parametric acoustic model: An acoustic decision tree for observation probability calculation," Proc. Interspeech2008, pp.289–292, Sept. 2008.
[5] R. Teunen and M. Akamine, "HMM-based speech recognition using decision trees instead of GMMs," Proc. Interspeech2007, pp.2097–2100, Sept. 2007.
[6] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, Classification and Regression Trees, Chapman & Hall, New York, 1984.
[7] S.J. Young, G. Evermann, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, The HTK Book, Cambridge University Engineering Department, UK, April 2005.
[8] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, 1993.
[9] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm," Machine Learning: Proceedings of the Thirteenth International Conference, pp.148–156, 1996.
[10] L. Breiman, "Bagging predictors," Mach. Learn., vol.24, no.2, pp.123–140, 1994.
[11] Y. Yuan and M.J. Shaw, "Induction of fuzzy decision trees," Fuzzy Sets and Systems, vol.69, no.2, pp.125–139, 1995.
[12] M. Riedmiller and H. Braun, "RPROP – Description and implementation details," Technical Report, Universitat Karlsruhe, 1994.
[13] H.G. Hirsch and D. Pearce, "The AURORA experimental framework for the performance evaluations of speech recognition systems

under noisy conditions," ISCA ITRW ASR2000.

Masami Akamine received the B.E. degree in electrical engineering from University of the Ryukyus in 1979 and the M.E. and the Ph.D. degrees in electrical engineering from Tohoku University in 1982 and 1985, respectively. Since 1985 he has been with the Corporate Research & Development Center, Toshiba Corporation. He is currently a senior fellow. His research interests include speech coding, speech synthesis, automatic speech recognition and their applications. He received the TELECOM System Technology Prize, the Technology Development Award from the Acoustical Society of Japan and the Society Paper Award from IEICE of Japan. He is a member of ASJ and a senior member of IEEE.

Jitendra Ajmera received the B.Tech. degree in electrical engineering from the Indian Institute of Technology (IIT), Bombay, in 1999 and the Ph.D. degree from EPFL, Lausanne, Switzerland, in 2004. Since 2005, he has worked at various research labs including the Corporate Research & Development Center, Toshiba Corporation. He is currently a researcher at IBM India Research Lab, New Delhi. His research interests include audio segmentation, speech recognition, text analysis and related applications.