HMM-based Speech Recognition Using Decision Trees Instead of GMMs

Remco Teunen, Masami Akamine

Toshiba Corporate Research & Development Center
1, Komukai-Toshiba-cho, Saiwai-ku, Kawasaki, 212-8582, Japan
masa.akamine@toshiba.co.jp

10.21437/Interspeech.2007-567

Abstract

In this paper, we experiment with decision trees 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 that they do not impose restrictions on the number or types of features, and that they automatically perform feature selection. In fact, due to the conditional nature of the decision tree evaluation process, the subset of features that is actually used during recognition depends on the input signal. Automatic state-tying can be incorporated directly into the acoustic model as well, and it too becomes a function of the input signal. 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.

Index Terms: speech recognition, acoustic modeling, decision trees, probability estimation, likelihood computation

1. Introduction

The acoustic model of a speech recognizer 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]. Decision trees (DTs) are not commonly used to perform this computation, although DTs have a number of interesting properties 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 efficiently be integrated.
- Automatic state-tying can be incorporated.
- The training process is inherently discriminative (at the frame-level) and performs automatic feature selection.
- Due to the conditional nature of the evaluation process of a DT, the subset of features that is actually used during recognition depends on the input signal.

Additionally, DTs can be interpreted relatively easily, especially if high-level features are used.

Our work is closely related to the work presented in [2], but our approach and focus are different. Instead of treating a decision tree as being merely a tree based vector quantizer, we view a decision tree as a tree based model with an integrated decision-making component. 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 that it can use to effectively specialize 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 (sub)models. The same idea applies to other types of questions as well, such as questions about the phonetic context, the acoustic environment and the speaker. In other words, our goal is to combine condition-specific models and state-tying directly into the DT acoustic model, and have all aspects of the model automatically be trained from data. Additionally, we propose the following techniques to improve the basic model: DT mixture models and soft decisions for continuous features.

The work presented in this paper is work in progress, since many (of the more advanced) aspects of DT acoustic models have not been explored yet. The goal of this paper is to lay the foundation for future work and to provide baseline performance numbers.

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 describes the experiments that have been conducted using the Aurora 2 speech database. Finally, Section 4 briefly describes our future plans and concludes this paper.

2. Decision tree acoustic models

DT acoustic models are HMM-based acoustic models that utilize decision trees instead of, for instance, GMMs or ANNs to compute observation likelihoods [2]. A decision tree determines the likelihood of an observation by asking a series of questions about the observation. 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. These types of DTs are also called *Probability Estimation Trees* (PETs).

Throughout this paper, we will assume that decision trees are implemented as binary trees. For a thorough treatment of decision trees, see [3] and [4].

2.1. Basic configuration

DTs can deal with multiple target classes at the same time and this makes it possible to use a single DT for all HMM states. However, we found that better results are obtained by using a different tree for each HMM state. In this configuration, each DT is trained to discriminate between the training data that corresponds to the associated HMM state ("positive" samples) and all other data ("negative" samples). The scaled likelihood $L(X_i \mid q_j)$ of observation X_i given state q_j can then be computed using

$$L(X_i \mid q_j) = \frac{P(q_j \mid X_i)}{P(q_j)}, \quad (1)$$

where $P(q_j \mid X_i)$ is the posterior probability of state q_j given observation X_i and $P(q_j)$ is the prior probability of state q_j.

The high-level training algorithm of DT acoustic models is similar to that of standard models. It consists of an initialization stage, followed by one or more stages that

change the complexity and/or structure of the model. Each stage is followed by several parameter re-estimation iterations using the Baum-Welch algorithm [1][2].

The initial model is created by using state-level forced alignments from an existing system, or by using a *flat start* approach [5]. The individual DT models are constructed using a fairly standard DT training algorithm. 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 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 positive samples. In order words, trees are grown to maximize the total (scaled) log likelihood of the positive training samples.

Once a tree is fully grown, the likelihood split criterion 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).

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][2]. 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.

A variation of the basic training algorithm is to grow DTs incrementally in stages, intermixed with Baum-Welch parameter re-estimation iterations. Although this approach was found to be better than using a single growing stage, it results in a significant increase in the total training time.

2.2. Automatic state-tying

An interesting extension to the basic DT acoustic model configuration is to allow DTs to perform automatic state-tying, or more precisely, *untying*. In this configuration, multiple HMM states share the same DT, tying the states together. However, the DT is allowed to ask questions about features that can identify the HMM state for which the model is evaluated, and hence it can untie states. We will 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 is very powerful and, for instance, also allows DTs to incorporate phonetic context-dependent state-tying directly into the acoustic model. Furthermore, the amount of state-tying can actually change depending on the input signal.

The training algorithm must treat decoding features differently from acoustic features, since the former are not part of the observation, but rather they are part of the HMM state identifier. As a result, the training algorithm deals with splits based on decoding features in the following way:

1. Negative samples are unaffected by the split and consequently both child nodes inherit all negative samples.
2. Positive samples are split according to the question, but the positive samples of each child are added to the other child as negative samples.

In short, both child nodes inherit *all* training data from the parent, but all non-corresponding positive samples are relabeled to be negative samples. As far as the training algorithm is concerned, splits based on decoding features do no actually split the training data, but only re-label positive samples based on the question.

The algorithm for evaluating a DT does not change, except that the observation likelihood is computed using the prior probability of the state or set of states that actually corresponds to each leaf.

2.3. Robustness

One unsatisfying aspect of the DT acoustic models described so far is that the likelihood as 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. Fortunately, several techniques are available to remedy this issue.

2.3.1. 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 [7] or bagging [8]. 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.

The down-side of using a method such as bagging is that the total number of model parameters increases linearly with the ensemble size. 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:

$$L(X_i \mid q_j) = \sum_{k=1}^{N} w_k L(X_i \mid \mathrm{M}_j^k) \,, \qquad (2)$$

where N is the number of components, w_k is the weight of component k, and M_j^k corresponds to the model of component k for state q_j. 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 retrain the components from scratch. This process is repeated for a certain number of iterations.

2.3.2. Soft decisions

Another method that can be used to improve the robustness of DTs is to make "soft" or "fuzzy" decisions, instead of hard decisions [9]. 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 sigmoidal function $f_d(x)$ given by

$$f_d(x) = \frac{1}{1 + \exp(s_d(t_d - x))}, \quad (3)$$

where t_d is the threshold of question 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_i given state q_j returned by (question) node d is given by

$$L_d(X_i \mid q_j) = f_d(x_{id}) L_{d0}(X_i \mid q_j) + (1 - f_d(x_{id})) L_{d1}(X_i \mid q_j), \quad (4)$$

where x_{id} is the feature value that node d asks about, $L_{d0}(X_i \mid q_j)$ is the likelihood that child 0 of node d returns and $L_{d1}(X_i \mid q_j)$ is the likelihood that child 1 of node d returns.

In our current implementation, we first train hard decision trees, convert them to soft decision trees and then re-estimate the t_d and s_d parameters using an iterative gradient-based optimization algorithm that tries to maximize the total training data likelihood. Our implementation is based on the RProp algorithm [11] 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 t_d and s_d parameter values.

The same optimization technique that is used to train the question parameters can be used for other purposes as well. For example, it should be possible to train feature transformations in a similar fashion, which means that techniques such as CMLLR [12] can be adapted for soft DT acoustic models. This path will be explored in future work.

Features that already represent a soft decision can be incorporated directly into to 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. Experimental results

All experimental results are based on the Aurora 2 speech database [10]. The task is connected digit string recognition under noisy conditions for the American-English language. This speech database was selected, because a) the task is a pure acoustic modeling problem, b) the baseline error rates are relatively high and c) 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).

All experiments were performed using a customized version of HTK v3.3.

3.1. Single DT per state

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 to 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 filterbank features plus their 1st, 2nd and 3rd derivatives. The total number of features is 68 in this case.

Table 1. *Average word accuracy of a standard GMM system and a comparable DT system, for two different feature sets.*

		Set A	Set B	Set C
GMM 1	[PLP + E]	**89.2%**	**88.6%**	**88.7%**
GMM 2	[PLP + FB8]	86.5%	86.5%	86.5%
DT 1	[PLP + E]	85.1%	81.7%	81.0%
DT 2	[PLP + FB8]	**89.2%**	**87.9%**	**89.1%**

The DT system uses a single tree per HMM state and the model complexity of each tree is the same as 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.

The results demonstrate 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. Due 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 filterbank features alleviates this problem and gives a significant boost in accuracy. Given that the base features that the DT system uses are PLP features, features that have been optimized for GMMs, it is not unlikely that there is a lot of room for improvements on the front-end side.

3.2. Mixture models

In the following experiment, mixtures of trees are used to model each HMM state. All mixtures have 3 components. 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. All models have the same number of parameters. 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.

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

	Set A	Set B	Set C
GMM 1	89.2%	88.6%	88.7%
DT 2, single	89.1%	87.8%	89.1%
DT 2, mixture 3	89.5%	88.5%	89.6%

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.

3.3. Soft decision trees

Another approach to improve the robustness of the base models is to use questions that make soft decisions instead of hard decisions. In this experiment, we take the hard decision model ("DT 2, single") and convert it to a soft decision model according to the method outlined in sub-subsection 2.3.2. For each (question) node d, the initial estimate of the smoothness parameter s_d is based on the variance of the feature at node d. After initialization of the smoothness parameters, all model parameters are re-estimated by running the optimization algorithm for 3 x 20 iterations. Note that the *structure* of the trees does not change.

Table 3. *Comparison of the word accuracy of a hard decision DT system and a soft decision DT system.*

	Set A	Set B	Set C
DT 2, hard decisions	89.1%	87.8%	89.1%
DT 2, soft decisions	91.2%	89.6%	90.9%

The results are shown in Table 3. The soft decision model is significantly better than the hard decision model (17% relative error rate reduction), but the accuracy improvement comes at a price: the number of model parameters is 50% higher and the model is computationally more expensive to train and evaluate. However, since the soft decision DT model is a smooth function that can be optimized using standard algorithms, it paves the way for other speech recognition techniques that are much more difficult to adapt to hard decision DT models, such as speaker adaptation and utterance-level discriminative training.

4. Conclusion and future work

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. However, the results are based on small complexity models and there is no guarantee the results will hold as the model complexity increases. On the other hand, the DT feature set has not been optimized yet and, in general, the algorithms are not as mature yet as those of the baseline GMM system.

Future work includes optimizing the feature set specifically for DT acoustic models and increasing the model complexity. We expect that we will be able to accomplish the latter by using soft decision DT mixture models. Once the DT system is competitive with a 20 Gaussian per mixture GMM system, higher-level information will be incorporated into the trees. Other future work includes utterance-level discriminative training and speaker adaptation using CMLLR.

5. References

[1] R. Cole et al. (eds), *Survey of the State of the Art in Human Language Technology*, Cambridge University Press, New York, 1997.

[2] J.T. Foote, *Decision-Tree Probability Modeling for HMM Speech Recognition*, Ph.D. thesis, Brown University, Providence, RI, 1993.

[3] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, *Classification and Regression Trees*, Chapman & Hall, New York, 1984.

[4] J.R. Quinlan, *C4.5: Programs for Machine Learning, Morgan Kaufmann*, San Mateo, CA, 1993.

[5] S.J. Young et al., *The HTK Book*, Cambridge, UK, April 2005.

[6] F. Jelinek, "*Continuous speech recognition by statistical methods*", Proceedings of the IEEE, vol. 64, no. 4, pp. 532-556, 1976.

[7] Y. Freund and R. Schapire, "*Experiments with a new boosting algorithm*", Machine Learning: Proceedings of the Thirteenth International Conference, pp. 148-156, 1996.

[8] L. Breiman, "*Bagging Predictors*", Machine Learning, Vol. 24, No. 2, pp. 123-140, 1994.

[9] Y. Yuan and M.J. Shaw, "*Induction of fuzzy decision trees*", Fuzzy Sets and Systems 69(2), pp. 125-139, 1995.

[10] H.G. Hirsch and D. Pearce, "*The AURORA Experimental Framework for the Performance Evaluations of Speech Recognition Systems under Noisy Conditions*", ISCA ITRW ASR2000.

[11] M. Riedmiller and H. Braun, "*RPROP -- Description and Implementation Details",* Technical Report, Universitat Karlsruhe, 1994.

[12] C. J. Leggetter and P. C. Woodland, "*Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models*," Comp. Spch. & Lang., vol. 9, pp. 171–185, Apr. 1995.