Modeling Phonetic Context with Non-random Forests for Speech Recognition

Hainan Xu[1], *Guoguo Chen*[1], *Daniel Povey*[1,2], *Sanjeev Khudanpur*[1,2]

[1]Center for Language and Speech Processing
[2]Human Language Technology Center of Excellence
The Johns Hopkins University, Baltimore, MD 21218, USA

hxu31@jhu.edu, guoguo@jhu.edu, dpovey@gmail.com, khudanpur@jhu.edu

Abstract

Modern speech recognition systems typically cluster triphone phonetic contexts using decision trees. In this paper we describe a way to build multiple complementary decision trees from the same data, for the purpose of system combination. We do this by jointly building the decision trees using an objective function that has an added entropy term to encourage diversity among the decision trees. After the trees are built, the systems are built in the standard way and the emission probabilities are combined during decoding. Experiments on multiple datasets show gains from the use of multiple trees, at the expense of evaluating multiple models in test time.

Index Terms: Decision Tree, Acoustic Modeling, Speech Recognition, Ensemble Methods

1. Introduction

While phones are widely used to represent word pronunciations, the so-called coarticulation effect means that the acoustic realization of a phone may depend on the phones to the right and left. Modern speech recognition systems typically use context dependent phones (e.g., triphones) as their modeling units. This, however, causes the problem of data sparsity, because most context dependent phones will not appear in the training data. Triphone contexts are normally clustered into equivalence classes, in order to reduce the number of distinct triphones we need to model and to handle unseen triphones.

The conventional technique to achieve these phonetic equivalence classes is the decision tree, in which the space of phonetic contexts is iteratively split by asking binary questions about the identities of the preceding and succeeding phones. This has the advantage of naturally giving reasonable equivalence classes for triphones that were never seen in the training data. While it is generally NP-hard to build an "optimal" decision tree [1], algorithms that efficiently build sub-optimal decision trees have been proved to be very successful in speech recognition. In [2], a greedy algorithm is proposed for building phonetic decision trees, where it greedily finds the "best split" on the current tree and splits the tree until a certain stopping criterion is reached. The objective function that we greedily optimized is normally based on modeling the feature data with a diagonal-covariance Gaussian model whose parameters are specific to each tree leaf.

Most speech recognition systems incorporate only a single decision tree to cluster context dependent phones, but a number of previous authors have looked into the possible benefits of building multiple trees, mostly for purposes of system combination. In [3] and [4], randomness is added to the tree building process to generate different but complementary phonetic decision trees. A separate acoustic model is trained for each tree; the system combination is based on (respectively) averaging the acoustic likelihoods; and ROVER [5]. In [6], a method called *factorized decision trees* is introduced, where different subsets of the data are used to build multiple phonetic decision trees. But unlike the work in [3] and [4], where acoustic models are trained on the whole training data with each decision tree, [6] trains acoustic models in a *Cluster Adaptive Training* framework [7]. In [8], multiple decision trees as well as multiple acoustic models are created by re-weighting the training data (as in boosting), and decoding is done using weighted-averaged log-likelihood scores.

Our work in this paper is similar to that in [3] and [4], where multiple decision trees are generated without splitting or modifying the training data. But instead of adding randomness to the tree building process, we propose a deterministic way of building multiple phonetic decision trees. We do this by introducing an entropy term to the tree building objective function, which encourages diversity among the trees. After the trees are built, the acoustic models are trained in the standard way and the combination is done by combining the emission probabilities from different models during decoding. Experiments on multiple datasets show consistent gains from the use of multiple trees.

The rest of the paper is structured as follows. In Section 2 we describe our objective function for building a collection of diverse decision trees (our "non-random forest"). In Section 3, we explain how we incorporate multiple decision trees into our speech recognition system. Details of our experimental setup are given in Section 4, and results are shown in Section 5. Finally in Section 6 we reiterate our main claims and discuss potential future work.

2. Multi-tree Algorithm

In this section, we first define entropy for single and multiple decision trees. We then introduce an entropy term to the standard phonetic decision tree building objective function, which encourages diversity when building multiple trees jointly from the same data. Finally we illustrate how we build multiple trees jointly.

2.1. Entropy of A Single Decision Tree

For each tree, we define a discrete probability distribution based on the data counts at each leaf. Let d be a decision tree with k leaves $1, 2, \ldots, k$, and $C(i)$ the count of training-data frames

This work was partially supported by NSF Grant No IIS 0963898, DARPA BOLT Contract No HR0011-12-C-0015, and an unrestricted gift from Google Inc. (No 2012-R2-106).

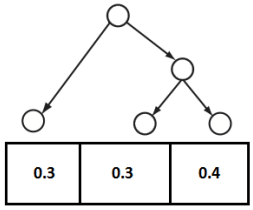

Figure 1: Distribution of a single decision tree

that are clustered to leaf i, we define the distribution as:

$$P_d(i) = \frac{C(i)}{N}, \text{for } 1 \leq i \leq k \tag{1}$$

where N is the number of frames in the dataset. Figure 1 illustrates a decision tree with three leaves and its corresponding probability distribution. The entropy of a decision tree can then be written as

$$H(d) = -\sum_{i=1}^{k} P_d(i)\log(P_d(i)) \tag{2}$$

Generally, the entropy of a decision tree defined above will have a larger value if the decision tree distributes the data to its leaves more evenly.

2.2. Joint Entropy of Multiple Decision Trees

The probability distribution and entropy defined above for the single decision tree case can easily be extended to the case of multiple decision trees, by using fraction of data at "joint" leaves. Suppose $D = \{d_1, d_2, ..., d_n\}$ is a set of decision trees, the joint probability distribution can be defined as follows

$$P_D(i_1, \ldots, i_n) = \frac{C(i_1, \ldots, i_n)}{N} \tag{3}$$

where $C(i_1, \ldots, i_n)$ is the number of data frames that appears in leaves i_j in j-th tree for all $j = 1, \ldots, n$. Figure 2 gives an example of the joint distribution of two trees, each with three leaves. Note that not all combinations of leaves of different trees have non-zero probabilities. Following the definition of the joint distribution, we can write the joint entropy of multiple decision trees as follows

$$H(D) = -\sum P_D(i_1, \ldots, i_n)\log(P_D(i_1, \ldots, i_n)) \tag{4}$$

where the summation is over all the possible combinations of $(i_1, \ldots, i_n)$'s. In all of the above entropy definitions, we follow the convention that $0 \log 0 = 0$.

2.3. Multi-tree Objective Function

In speech recognition, greedy algorithms are usually used to build phonetic decision trees. In such algorithms, one tree split is performed each time, optimizing some pre-defined objective functions, and the splitting goes on until certain criteria is reached. A commonly used objective function is the training data likelihood. For example, in [2], the Gaussian likelihood is used as the objective function for building trees, assuming data is generated by Gaussian distributions whose parameters only depends on the leaf it is assigned to. Note that maximizing the total likelihood is equivalent to maximizing the average likelihood per data frame, and in this paper we define the objective functions as likelihood per data frame.

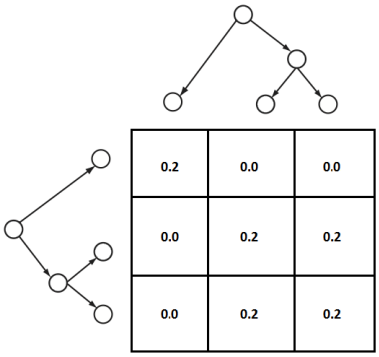

Figure 2: Joint distribution of multiple decision trees

Let $L(d)$ be the averaged Gaussian likelihood of data on phonetic decision tree d, the objective function to maximize during the decision tree building as described in [2] is equivalent to

$$\mathcal{F}(d) = L(d) \tag{5}$$

Now suppose we want to build a list of decision trees $D = \{d_1, \ldots, d_n\}$ jointly. Let $L(D)$ be the data likelihood from all the decision trees, i.e.

$$L(D) = \sum_{i=1}^{n} L(d_i). \tag{6}$$

We use the following objective function for building multiple trees:

$$\mathcal{F}(D) = L(D) + \lambda \left(H(D) - \frac{\sum_{i=1}^{n} H(d_i)}{n} \right) \tag{7}$$

where $H(D)$ is the joint entropy of multiple decision trees D, and $H(d_i)$ is the entropy of the individual decision tree d_i.

This new objective function is equivalent to (5) when building a single decision tree since the joint entropy equals to the single tree entropy when there is only one tree. If there are multiple trees, the added term will encourage diversity among the trees– the larger λ is, the stronger this effect will be.

It is worth mentioning that adding the $H(D)$ term alone is sufficient to encourage diversity among the trees. But a side effect of adding that term only is that the system tends to make the individual trees' entropies large as well, by splitting leaves with large counts of data. By adding the negated term in (7), we counteract this effect.

2.4. Multi-tree Algorithm

Our tree building process is very similar to the standard algorithm described in [2], except that we build multiple trees jointly instead of one, and we also use the objective function in Equation (7) instead of that in Equation (5).

We will not be describing the exact data structures we used to do the multi-tree splitting. The key thing is that our data structures need to make it possible to to efficiently compute the change in the entropy term from splitting one of the trees.

We start the tree building process by initializing monophone trees for the specified number of decision trees. Then, each time, we split one of the decision trees with the split that maximizes the change in Equation (7). We repeat this process until all decision trees obtain the required number of leaves. After the splitting is done for all the trees, we also do the merging

described in [2], which merges leaves within a tree if the decrease of the objective function caused by the merge is less than the smallest gain during the splitting step. Therefore, the final decision trees will have fewer leaves than the number initially obtained in the splitting phase.

3. Application of Multi-tree Method to Speech Recognition

This section describe how we incorporate the decision trees built in Section 2 into our speech recognition systems.

3.1. Acoustic Model Training

Acoustic models are trained in the standard way, the same as those in the single tree case. But we build one model with each decision tree in the multi-tree case.

3.2. Decoding

There are different ways to combine individual acoustic models trained with each each decision tree. For example, we can decode with each acoustic model independently, and do system combination on the decoding output. In this work, we combine at the state level, by computing the combined data likelihoods $p(o|s)$. (Note that in the DNN case we deal with pseudo-likelihoods and not real likelihoods).

We follow the work in [9] on *tree arrays*, and build a "virtual tree", such that every leaf in the virtual tree corresponds to a unique combinations of leaves in individual trees. Let s be a leaf (which also corresponds to a state in the hidden Markov model) in the virtual tree, and let $s_{1j_1}, s_{2j_2}, \ldots, s_{nj_n}$ be its corresponding leaves from the individual trees, where s_{ij_i} is the j_i^{th} leaf of the i^{th} tree. An obvious way to do combination is to take the geometric mean of the likelihoods (equivalently, the algebraic mean of the log-likelihoods). In [3], the algebraic mean of the likelihoods was used.

In this work, we use a method which we found experimentally to work better than either of the above-mentioned methods:

$$\log(p(o|s)) = \frac{\sum_{i=1}^{n} \log(p(o|s_{ij_i})) \exp(C \log(p(o|s_{ij_i})))}{\sum_{i=1}^{n} \exp(C \log(p(o|s_{ij_i})))}. \tag{8}$$

where for experiments reported here, we set $C = 0.1$.

As for transition probabilities, we simply take the algebraic means of the transition probabilities given by each model.

4. Experimental Details

We use the open-source speech recognition toolkit Kaldi [10] for our implementation and experiments.

4.1. System Description

We start off by training a speaker-independent Gaussian mixture model (GMM) system, which will later be used for both the baseline system and our proposed system. For this GMM system, we extract 13 dimensional Mel-frequency cepstral coefficient (MFCC) [11] features, and perform a typical maximum likelihood acoustic training recipe that begins with a flat-start initialization of context-independent phonetic HMMs, and ends with learning the maximum likelihood linear transform (MLLT) [12] for features.

For the baseline system, we move forward by re-building the decision tree with statistics collected from the speaker-independent GMM system, and performing speaker adaptive training (SAT). This is further followed by a deep neural network (DNN) training with p-norm nonlinearity [13]. Both the GMM and DNN baseline systems are tuned so that increasing the number of parameters would result in degradation of performances.

For our proposed system, we use the same statistics collected from the speaker-independent GMM system, but we build multiple decision trees with the algorithm described in Section 2. For each decision tree, we build a SAT model, which is then followed by the p-norm DNN training. Therefore, in our proposed system, we have multiple acoustic models in both the SAT and DNN stage. However, we are careful not to perform further re-alignment of the data after building multiple trees. We need the alignments to remain compatible for the joint decoding to make sense (i.e. if the different systems learned to align data in incompatible ways, it would be a problem).

4.2. Datasets

We conduct experiments on four datasets, namely *Wall Street Journal (WSJ)* [14], *Switchboard (SWBD)*, *TED-LIUM* [15] and *Librispeech* [16].

For *Wall Street Journal*, acoustic models are trained on the SI-286 portion of the training data, and a trigram language model is used for decoding; performance is then evaluated on the eval92 and dev93 datasets. For *Switchboard*, acoustic models are trained on the whole training set, and 4gram language model trained from *Fisher* data is used for decoding; we evaluate performance on the full eval2000 dataset as well as its *SWBD* subset. For *Librispeech*, acoustic models are trained on the 100 hours subset, and a 4gram language model is used for decoding; we evaluate on both "dev" and "test" datasets under clean and noisy ("other") conditions. For *TED-LIUM*, acoustic models are trained on the whole training data, and trigram language model is used for decoding; the performance is then evaluated on both "dev" and "test" datasets.

5. Results

5.1. Impact of the Entropy Term

# trees	λ	avg-entropy	joint-entropy
1	-	7.63	7.63
2	0.1	7.67	7.85
2	0.25	7.72	8.11
2	0.5	7.76	8.41
2	1	7.78	8.78
3	1	7.74	9.00
4	1	7.72	9.07

Table 1: Entropy of multi-trees (*TED-LIUM*)

Table 1 shows the average entropy of the trees, together with the joint entropy, for multiple decision trees built from the same *TED-LIUM* data, with varying values of λ. The stopping criterion for the splitting is when each tree reaches 5000 leaves, but the actual number of leaves per tree is less than 5000 after the merge operation described in Section 2.

From Table 1 we can see that if we increase the weight of the entropy term (λ), the average entropy of multiple trees only changes slightly, but the joint entropy of all the trees increases significantly. This is because the entropy term is moving the decision trees away from each other, and increasing the entropy

# trees	λ	avg # leaves	# virtual-leaves
1	-	3973	3973
2	0.1	4030	8173
2	0.25	4115	12969
2	0.5	4204	21138
2	1	4237.5	36828
3	1	4123	97999
4	1	4078.5	164811

Table 2: Number of leaves in multi-trees (*TED-LIUM*)

term weight λ makes the trees more different. Table 2 displays the average number of leaves in the individual trees, together with the number of virtual leaves, showing the same trend that larger λ increases diversity among the trees.

5.2. Impact of λ on WER

Knowing that increasing λ will lead to more diversity in the decision trees, we run another set of experiments to evaluate the impact of λ on WER. Table 3 shows WER performance of the multi-tree systems with different values of λ on the SAT model of *TED-LIUM* corpus. We can see that by introducing diversity among the trees, it helps improve the recognition performances; however if λ gets too large, performance of the combination system starts to degrade, because with large λ the individual models get worse, as we will see in Section 5.3. For the rest of our experiments, we will use $\lambda = 1$ since it gives the most improvement on *TED-LIUM* corpus, as shown in Table 3.

# trees	λ	dev	test
1	-	25.8	23.5
2	0.1	25.4	23.2
	0.5	25.2	22.9
	1	25.1	**22.7**
	1.5	**25.0**	23.0
	2	25.2	22.9

Table 3: WER of SAT models on *TED-LIUM*

5.3. Comparison of Different Combination Methods

Table 4 compares the following results, all based on DNN models:

baseline	The baseline system
tree 1	Two-tree system: first tree by itself
tree 2	Two-tree system: second tree by itself
MBR	Two-tree system: multi-lattice *Minimum Bayes Risk* decoding as in [17]
joint	Two-tree system: proposed joint decoding method

The individual trees by themselves perform worse than the baseline, as expected. The combination based on MBR decoding is a little better than the individual models, but still worse than the baseline. However our joint-decoding system substantially outperforms both the individual trees and the baseline.

	dev		test	
# trees	clean	other	clean	other
baseline	5.93	20.42	6.59	22.47
tree 1	6.20	20.67	6.75	22.68
tree 2	6.27	21.07	6.87	22.84
MBR	6.00	20.87	6.59	22.84
joint	**5.82**	**19.86**	**6.46**	**21.62**

Table 4: WER of individual and combined DNN models on *Librispeech* ($\lambda = 1$)

5.4. Performance of Multi-tree DNN models

	WSJ		SWBD		TED-LIUM	
# trees	eval92	dev93	swbd	eval2000	dev	test
1	7.07	4.06	13.4	19.2	21.7	19.4
2	6.55	4.08	13.0	18.8	**21.2**	18.6
3	**6.46**	**3.72**	**12.8**	**18.7**	**21.2**	**18.5**

Table 5: WER of DNN models on *WSJ, SWBD* and *TED-LIUM* ($\lambda = 1$)

	dev		test	
# trees	clean	other	clean	other
1	5.93	20.42	6.59	22.47
2	5.82	19.86	6.46	**21.62**
3	**5.80**	**19.77**	**6.27**	21.68

Table 6: WER of DNN models on *Librispeech* ($\lambda = 1$)

Tables 5 and 6 give the multi-tree performance of the DNN models. As we can see from the tables, 2-tree systems give better recognition accuracy compared to the baseline consistently across most datasets; 3-tree systems usually outperform 2-tree systems, though the relative gains become smaller. We would expect further but relatively smaller improvements from using more trees. A downside of the multi-tree combination systems is that both training and decoding take longer, which is linear to the number of trees in the system. Therefore in our experiments we use at most 3 trees.

6. Conclusions and Future Work

In this paper, we present a method to build multiple phonetic decision trees for speech recognition. An entropy term is added to the standard objective function during tree building so that trees built jointly will grow differently. Acoustic models are built on top of different trees, and during decoding, emission probabilities from individual models are combined to give the final acoustic scores. Experiments on multiple datasets show that using multiple trees gives consistent gains for speech recognition.

For future work, we would first study other ways to combine different ASR systems besides combining acoustic scores at the state level; secondly, we want to compare the performance of this work with other multi-tree methods, e.g. randomized decision trees; thirdly, for DNN multi-tree systems, we would like to try letting the different models share parameters for the early layers, which would enable us to share most of the computation both when training and decoding.

7. References

[1] L. Hyafil and R. L. Rivest, "Constructing optimal binary decision trees is np-complete," *Information Processing Letters*, vol. 5, no. 1, pp. 15–17, 1976.

[2] S. J. Young, J. J. Odell, and P. C. Woodland, "Tree-based state tying for high accuracy acoustic modelling," in *Proceedings of the Workshop on Human Language Technology*, ser. HLT '94. Stroudsburg, PA, USA: Association for Computational Linguistics, 1994, pp. 307–312.

[3] J. Xue and Y. Zhao, "Random forests of phonetic decision trees for acoustic modeling in conversational speech recognition," *Proceedings of IEEE Transactions on Audio, Speech, and Language*, vol. 16, no. 3, pp. 519–528, March 2008.

[4] O. Siohan, B. Ramabhadran, and B. Kingsbury, "Constructing ensembles of asr systems using randomized decision trees," in *Proc. of International Conference on Acoustics Speech and Signal Processing*, 2005.

[5] J. G. Fiscus, "A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover)," in *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*. IEEE, 1997, pp. 347–354.

[6] K. Yu and H. Xu, "Cluster adaptive training with factorized decision trees for speech recognition," in *Proceedings of INTERSPEECH*. ISCA, 2013, pp. 1243–1247.

[7] M. J. Gales, "Cluster adaptive training of hidden markov models," *Speech and Audio Processing, IEEE Transactions on*, vol. 8, no. 4, pp. 417–428, 2000.

[8] G. Saon and H. Soltau, "Boosting systems for large vocabulary continuous speech recognition," *Speech communication*, vol. 54, no. 2, pp. 212–218, 2012.

[9] H. Soltau, G. Saon, and B. Kingsbury, "The ibm attila speech recognition toolkit," in *Spoken Language Technology Workshop (SLT), 2010 IEEE*. IEEE, 2010, pp. 97–102.

[10] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlíček, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," 2011.

[11] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 28, no. 4, pp. 357–366, 1980.

[12] M. J. Gales, "Semi-tied covariance matrices for hidden Markov models," *Speech and Audio Processing, IEEE Transactions on*, vol. 7, no. 3, pp. 272–281, 1999.

[13] X. Zhang, J. Trmal, D. Povey, and S. Khudanpur, "Improving deep neural network acoustic models using generalized maxout networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 215–219.

[14] D. B. Paul and J. M. Baker, "The design for the wall street journal-based csr corpus," in *Proceedings of the Workshop on Speech and Natural Language*, ser. HLT '91. Stroudsburg, PA, USA: Association for Computational Linguistics, 1992, pp. 357–362.

[15] A. Rousseau, P. Deléglise, and Y. Estève, "Enhancing the ted-lium corpus with selected data for language modeling and more ted talks," in *Proc. of LREC*, 2014, pp. 3935–3939.

[16] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an ASR corpus based on public domain audio books," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015.

[17] H. Xu, D. Povey, L. Mangu, and J. Zhu, "Minimum bayes risk decoding and system combination based on a recursion for edit distance," *Computer Speech & Language*, vol. 25, no. 4, pp. 802–828, 2011.