Modeling Phonetic Context with Non-random Forests for Speech Recognition

Hainan Xu

Center for Language and Speech Processing, Johns Hopkins University

September 4, 2015
In this presentation, we will

- Give a very brief introduction of how decision trees are used in the standard automatic speech recognition frameworks.
- Present our method that generalizes from using one decision tree to multiple trees (the non-random forest) which is combined with ensemble methods to improve recognition accuracy.
Till this day, the *Hidden Markov Model* is still the most successful model for speech recognition.

The 3-state *HMM* assumes there are “stages” of acoustic realization of phones.

Its parameters include transition probabilities $p(s'|s)$ and emission “probabilities” $p(o|s)$. 
We have a 3-state HMM for each “phone”.
We concatenate the phone models as word models.
Word models are connected according to the structure of a language model to form a decoding graph.
In decoding, given the acoustic observation, we find the most likely sequence in the graph.
“Phones” as ASR modeling units

It’s simple then. We have a model for each IPA symbol, such as t, b, h, ay, etc... TA-DA!

Not so easy!
t in “better” and “return”
t in “teacher” and “mountain”
r in “car” and “ray”

And then, linguistics came up with allophones.

But that helped just a little bit. Same allophones might still look very different in spectrograms.
“Phones” as ASR modeling units

- “It’s simple then. We have a model for each IPA symbol, such as t, b, h, ay, etc... TA-DA!”
“Phones” as ASR modeling units

- “It’s simple then. We have a model for each IPA symbol, such as t, b, h, ay, etc... TA-DA!”
- Not so easy!
“Phones” as ASR modeling units

- “It’s simple then. We have a model for each IPA symbol, such as t, b, h, ay, etc… TA-DA!”
- Not so easy!
  - t in “better” and “return”
“Phones” as ASR modeling units

- “It’s simple then. We have a model for each IPA symbol, such as t, b, h, ay, etc... TA-DA!”
- Not so easy!
  - t in “better” and “return”
  - t in “teacher” and “mountain”
“Phones” as ASR modeling units

- “It’s simple then. We have a model for each IPA symbol, such as t, b, h, ay, etc... TA-DA!”
- Not so easy!
  - t in “better” and “return”
  - t in “teacher” and “mountain”
  - r in “car” and “ray”
"Phones" as ASR modeling units

- "It’s simple then. We have a model for each IPA symbol, such as t, b, h, ay, etc... TA-DA!"
- Not so easy!
  - t in "better" and "return"
  - t in "teacher" and "mountain"
  - r in "car" and "ray"
- And then, linguistics came up with allophones.
“Phones” as ASR modeling units

• “It’s simple then. We have a model for each IPA symbol, such as t, b, h, ay, etc... TA-DA!”

• Not so easy!
  • t in “better” and “return”
  • t in “teacher” and “mountain”
  • r in “car” and “ray”

• And then, linguistics came up with allophones.

• But that helped just a little bit. Same allophones might still look very different in spectrograms.
Context Matters!

- The realizations of “w” varies but similar patterns occur with the same context.
- Instead of using phones regardless of their contexts (monophones), we usually consider the phone to the left and to the right.
- A phone in such context is called a triphone.
Triphones

- trees: SIL-tr+ee tr-ee+z ee-z+SIL
- better: SIL-b+eh b-eh+t eh-t+er t-er+SIL
Triphone

- Triphones
  - trees: SIL-tr+ee tr-ee+z ee-z+SIL
  - better: SIL-b+eh b-eh+t eh-t+er t-er+SIL
- There are around 50 phones in English.
Triphone

- Triphones
  - trees: SIL-tr+ee tr-ee+z ee-z+SIL
  - better: SIL-b+eh b-eh+t eh-t+er t-er+SIL
- There are around 50 phones in English.
- Thus around $50^3$ triphones.
  - We need $50^3 \times 3$ HMM states
  - Some triphones are rarely, if at all, seen in training data, but still require a model, which is a problem.
Triphone

- Triphones
  - trees: SIL-tr+ee tr-ee+z ee-z+SIL
  - better: SIL-b+eh b-eh+t eh-t+er t-er+SIL

- There are around 50 phones in English.
- Thus around $50^3$ triphones.
  - We need $50^3 \times 3$ HMM states
  - Some triphones are rarely, if at all, seen in training data, but still require a model, which is a problem.

- Solution: use decisions tree to create equivalence classes as units for parameter sharing.
Decision Trees

picture from http://speechlab.sjtu.edu.cn/ kyu/sites/kyu/files/teaching/Lecture02.pdf
Key Factors for Decision Tree Building

- A set of questions
- An objective function to maximize, $\mathcal{F}([\text{partial}] \text{ tree})$
  - Then we could define the “gain” of a split or the “cost” of a merge.
- A stopping criteria
- Algorithm for growing the tree
  - Getting the “optimal” decision tree is NP-hard.
  - Usually a greedy algorithm is used, i.e. keep splitting the tree with the “best question” until stopping criteria is met.
Questions are in the form of “is the previous phone in the set \{m, n, ng\}?”
Usually we want the questions to be “complete”, meaning there is a single element set for each phone which you could use in a question.

The objective function is usually the Gaussian likelihood of all data, assuming data mapped to the same leaf is generated from a Gaussian distribution.

We predefine the number of leaves we want in the tree as the stopping criteria.

In speech recognition, the most used tree-building algorithm is from Steve Young’s paper “Tree-based state tying for high accuracy acoustic modelling”.

Single-tree Building Algorithm, Steve Young et al.

1. initialize a monophone state tree
2. \( n = \) number of leaves we want
3. \textbf{while} we have \(< n \) leaves \textbf{do}
4. \( s = \) the best split on the current tree
5. apply split \( s \)
6. \textbf{end while}
Building Single Phonetic Decision Tree - Algorithm

Single-tree Building Algorithm, Steve Young et al.

1: initialize a monophone state tree
2: \( n = \) number of leaves we want
3: while we have \(< n\) leaves do
4: \( s = \) the best split on the current tree
5: apply split \( s\)
6: end while
7: while true do
8: \( c = \) the smallest cost of a merge between any leaves in the current tree
9: if \( c < \) merge threshold then
10: apply the merge corresponding to \( c\)
11: else
12: terminate and return the tree
13: end if
14: end while
The tree provides a mapping from a triphone state to one of its leaves. Based on the mapping, we train an acoustic model that could compute

$$\log p(\text{observation} | \text{triphone state}) = \log p(\text{observation} | \text{leaf in the tree})$$

Now with number of leaves $<<$ number of all possible triphones, acoustic model training is possible.

We build a graph which incorporates the information of leaves in the tree for decoding.
However, a single decision tree might be biased.
However, a single decision tree might be biased.

- Not optimal since we use a greedy algorithm.
- The objective function for growing the tree might not be the ground truth goodness function (remember we used Gaussian).
- ...
Extend to Multiple Trees

- However, a single decision tree might be biased.
  - Not optimal since we use a greedy algorithm.
  - The objective function for growing the tree might not be the ground truth goodness function (remember we used Gaussian).
  - ...

- We want to build different trees, and (hopefully) compensate for the bias by somehow combining the trees.
Extend to Multiple Trees

- However, a single decision tree might be biased.
  - Not optimal since we use a greedy algorithm.
  - The objective function for growing the tree might not be the ground truth goodness function (remember we used Gaussian).
  - ...

- We want to build different trees, and (hopefully) compensate for the bias by somehow combining the trees.

- Questions, stopping criteria: no change required.

- Objective function, algorithm: there is a problem since they’re deterministic, and will build exactly same trees.
Extend to Multiple Trees

- However, a single decision tree might be biased.
  - Not optimal since we use a greedy algorithm.
  - The objective function for growing the tree might not be the ground truth goodness function (remember we used Gaussian).
  - ...

- We want to build different trees, and (hopefully) compensate for the bias by somehow combining the trees.

- Questions, stopping criteria: no change required.

- Objective function, algorithm: there is a problem since they’re deterministic, and will build exactly same trees.

- Our solution: to include an “entropy term” in the objective function for tree-building.
A decision tree defines a “distribution”/partition on the data.

In the example above, the entropy is

\[-0.3 \log 0.3 - 0.3 \log 0.3 - 0.4 \log 0.4\]
A decision tree defines a “distribution” / partition on the data.

In the example above, the entropy is

$$-0.3 \log 0.3 - 0.3 \log 0.3 - 0.4 \log 0.4$$
(Joint-)Entropy of Multiple Decision Trees

- Multiple \( n \) decision trees split the data into an \( n \)-dimension grid.

- Note: not all combinations are possible.
- Also, not all possible combinations exist in data.
- The entropy of the above example is

(Joint-)Entropy of Multiple Decision Trees

- Multiple \( n \) decision trees split the data into an \( n \)-dimension grid.

- Note: not all combinations are possible.
- Also, not all possible combinations exist in data.
- The entropy of the above example is

\[
-0.2 \log 0.2 - 0.2 \log 0.2 - 0.2 \log 0.2 - 0.2 \log 0.2 - 0.2 \log 0.2
\]
We have an objective function defined on single tree as $G(\text{tree})$. We have introduced entropy of tree[s], noted as $H(\text{tree}[s])$. For multiple trees, we define the new objective function as:

$$\lambda \left( H(\text{all trees}) - \sum_i H(i\text{th tree}) \right) + \sum_i G(i\text{th tree})$$

The first 2 terms could push the joint-entropy to grow larger, while keeping the single tree entropies smaller, making sure the trees are different.
Multiple Tree - the New Objective Function

- We have an objective function defined on single tree as $G(\text{tree})$.
- We have introduced entropy of tree[s], noted as $H(\text{tree[s]})$.
- For multiple trees, we define the new objective function as,

$$
\lambda \left( H(\text{all trees}) - \frac{\sum_i H(\text{ith tree})}{n} \right) + \sum_i G(\text{ith tree})
$$

- The first 2 terms could push the joint-entropy to grow larger, while keeping the single tree entropies smaller, making sure the trees are different.
Building Multiple Trees - Algorithm

Multi-tree Building Algorithm

1: initialize a set of monophone state trees
2: \( n \) = number of leaves per tree that we want
3: while not all trees have \( n \) leaves do
4: \( s \) = the best split on trees having < \( n \) leaves
5: apply split \( s \)
6: end while
Building Multiple Trees - Algorithm

Multi-tree Building Algorithm

1: initialize a set of monophone state trees
2: \( n \) = number of leaves per tree that we want
3: while not all trees have \( n \) leaves do
4: \( s \) = the best split on trees having \(< n \) leaves
5: apply split \( s \)
6: end while
7: while true do
8: \( c \) = the smallest cost of a merge between any leaves in a same tree
9: if \( c < \) merge threshold then
10: apply the merge corresponding to \( c \)
11: else
12: terminate and return the trees
13: end if
14: end while
- Acoustic models are independently trained on top of each tree.
- The independent trainings ensure that the multi-tree methods could work regardless the type of acoustic model used (GMM or DNN or LSTM).
- There are different ways to combine the dependently trained acoustic models.
  - Our method: to merge acoustic likelihoods.
Our method tries to combine $p(\text{observation}|\text{triphone state})$.
For the same $p(o|s)$, each model would give a different likelihood score, in log-likelihoods $\{l_i\}$.
We use a “weighted” average that favors the larger likelihood scores (larger probabilities).

$$\tilde{l} = \frac{\sum_i l_i \exp(C \cdot l_i)}{\sum_i \exp(C \cdot l_i)}, \quad C = \text{acoustic scale} = 0.1$$

Transition probabilities in HMM are relatively much less important and we simply use the arithmetic means of the transition probabilities from each model.
A “virtual tree” is built such that each leaf in the virtual tree corresponds to a unique and valid combination of leaves from individual trees. It might have \( \geq 5 \) leaves in the virtual tree.

The virtual tree leaves correspond to the de facto parameter sharing units.

The virtual tree will be used in building decoding graphs.
Impact of the added Entropy Term

- This table shows that the added entropy term creates much finer parameter sharing units, i.e. having large number of virtual leaves.
- The stopping criteria is when each tree reaches 5000 leaves. The average number is smaller because of merging.

<table>
<thead>
<tr>
<th># trees</th>
<th>$\lambda$</th>
<th>avg # leaves</th>
<th># virtual-leaves</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>3973</td>
<td>3973</td>
</tr>
<tr>
<td>2</td>
<td>0.1</td>
<td>4030</td>
<td>8173</td>
</tr>
<tr>
<td>2</td>
<td>0.25</td>
<td>4115</td>
<td>12969</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>4204</td>
<td>21138</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>4237.5</td>
<td>36828</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>4123</td>
<td>97999</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4078.5</td>
<td>164811</td>
</tr>
</tbody>
</table>

**Table:** Number of leaves in multi-trees (*TED-LIUM*)
This table shows that the added entropy term does increase the joint-entropy while not increasing single tree entropies too much.

<table>
<thead>
<tr>
<th># trees</th>
<th>λ</th>
<th>avg-entropy</th>
<th>joint-entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>7.63</td>
<td>7.63</td>
</tr>
<tr>
<td>2</td>
<td>0.1</td>
<td>7.67</td>
<td>7.85</td>
</tr>
<tr>
<td>2</td>
<td>0.25</td>
<td>7.72</td>
<td>8.11</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>7.76</td>
<td>8.41</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>7.78</td>
<td>8.78</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>7.74</td>
<td>9.00</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>7.72</td>
<td>9.07</td>
</tr>
</tbody>
</table>

Table: Entropy of multi-trees (TED-LIUM)
Performance of Multi-tree Decoding

- Here we compare the recognition results by decoding each individual model, the MBR decoding and our “joint”-decoding method.

<table>
<thead>
<tr>
<th># trees</th>
<th>clean</th>
<th>other</th>
<th>clean</th>
<th>other</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>5.93</td>
<td>20.42</td>
<td>6.59</td>
<td>22.47</td>
</tr>
<tr>
<td>tree 1</td>
<td>6.20</td>
<td>20.67</td>
<td>6.75</td>
<td>22.68</td>
</tr>
<tr>
<td>tree 2</td>
<td>6.27</td>
<td>21.07</td>
<td>6.87</td>
<td>22.84</td>
</tr>
<tr>
<td>MBR</td>
<td>6.00</td>
<td>20.87</td>
<td>6.59</td>
<td>22.84</td>
</tr>
<tr>
<td>joint</td>
<td><strong>5.82</strong></td>
<td><strong>19.86</strong></td>
<td><strong>6.46</strong></td>
<td><strong>21.62</strong></td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th># trees</th>
<th>eval92</th>
<th>dev93</th>
<th>swbd</th>
<th>eval2000</th>
<th>dev</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7.07</td>
<td>4.06</td>
<td>13.4</td>
<td>19.2</td>
<td>21.7</td>
<td>19.4</td>
</tr>
<tr>
<td>2</td>
<td>6.55</td>
<td>4.08</td>
<td>13.0</td>
<td>18.8</td>
<td>21.2</td>
<td>18.6</td>
</tr>
<tr>
<td>3</td>
<td>6.46</td>
<td>3.72</td>
<td>12.8</td>
<td>18.7</td>
<td>21.2</td>
<td>18.5</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th># trees</th>
<th>clean</th>
<th>other</th>
<th>clean</th>
<th>other</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.93</td>
<td>20.42</td>
<td>6.59</td>
<td>22.47</td>
</tr>
<tr>
<td>2</td>
<td>5.82</td>
<td>19.86</td>
<td>6.46</td>
<td><strong>21.62</strong></td>
</tr>
<tr>
<td>3</td>
<td><strong>5.80</strong></td>
<td><strong>19.77</strong></td>
<td><strong>6.27</strong></td>
<td>21.68</td>
</tr>
</tbody>
</table>
Conclusion

- Multi-tree systems could consistently give better results than single tree systems.
- Multi-tree systems are especially helpful for speech recordings with noisy backgrounds.
- The more trees we use, the more it helps; though the gain becomes much smaller for larger numbers.