Modeling Phonetic Context with Non-random Forests for Speech Recognition

Hainan Xu

Center for Language and Speech Processing,
Johns Hopkins University

September 4, 2015

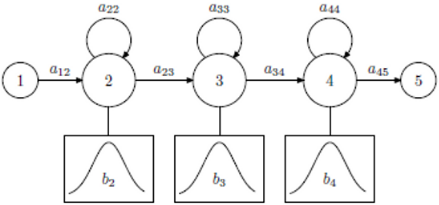
Outline

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 deision tree to multiple trees (the non-random forest) which is combined with ensemble methods to improve recognition accuracy.

Speech Recognition 101, *HMMs*

- 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" $p(o|s)$.

HMM for Speech Recognition

- 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

"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"

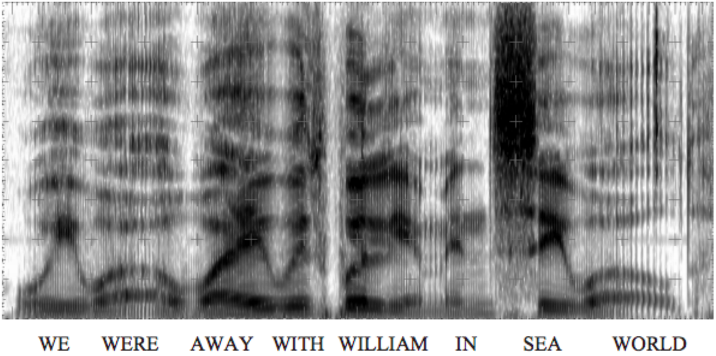
"Phones" as ASR modeling units

- "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.

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

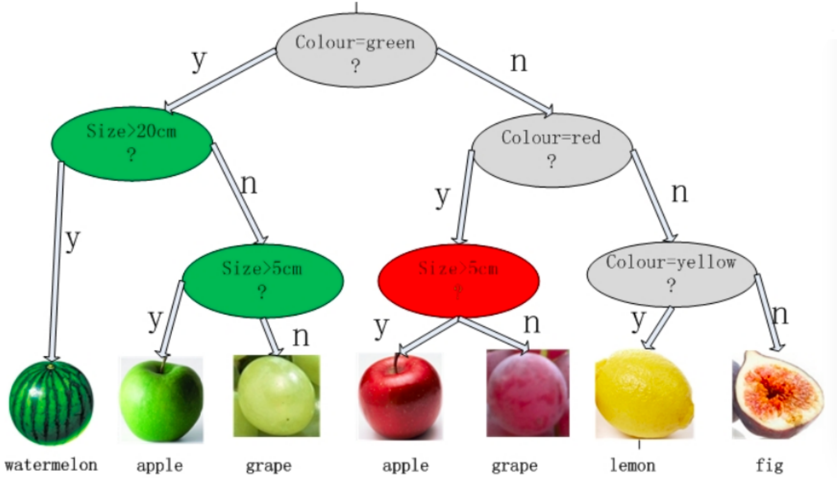
- Triphones
  - trees: SIL-tr+ee tr-ee+z ee-z+SIL
  - better: SIL-b+eh b-eh+t eh-t+er t-er+SIL
- There are around 50 phones in English.

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 $50^3 * 3$ *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 $50^3 * 3$ *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.



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, $\mathcal{F}$([partial] 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.

Phonetic Decision Trees

- 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".

Building Single Phonetic Decision Tree - Algorithm

Single-tree Building Algorithm, Steve Young et al.

initialize a monophone state tree
$n =$ number of leaves we want
while we have $< n$ leaves **do**
    $s =$ the best split on the current tree
    apply split s
end while

Building Single Phonetic Decision Tree - Algorithm

Single-tree Building Algorithm, Steve Young et al.

```
initialize a monophone state tree
n = number of leaves we want
while we have < n leaves do
    s = the best split on the current tree
    apply split s
end while
while true do
    c = the smallest cost of a merge between any leaves in the current tree
    if c < merge threshold then
        apply the merge corresponding to c
    else
        terminate and return the tree
    end if
end while
```

Models on Top of the Tree

- 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.

Extend to Multiple Trees

- However, a single decision tree might be biased.

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).
  - ...

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

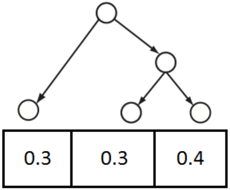
- However, a single decision tree might be biased.
  - Not optimal since we use a greedy algorithm.
  - The objective function for growing the tree might not be the ground truth goodness function (remember we used Gaussian).
  - ...
- We want to build different trees, and (hopefully) compensate for the 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.

Entropy of a Decision Tree

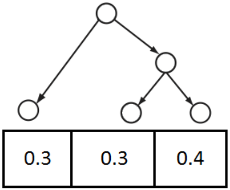
- A decision tree defines a "distribution"/partition on the data.



- In the example above, the entropy is

Entropy of a Decision Tree

- A decision tree defines a "distribution"/partition on the data.

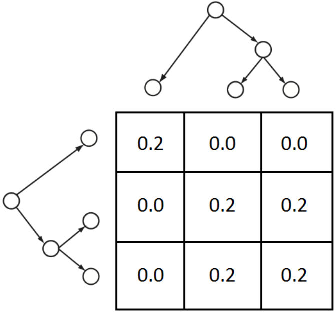


- In the example above, the entropy is

$$-0.3 \log 0.3 - 0.3 \log 0.3 - 0.4 \log 0.4$$

(Joint-)Entropy of Multiple Decision Trees

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

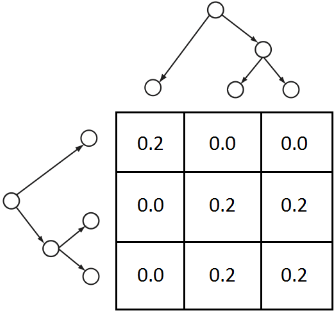


0.2	0.0	0.0
0.0	0.2	0.2
0.0	0.2	0.2

- 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

$$-0.2 \log 0.2 - 0.2 \log 0.2 - 0.2 \log 0.2 - 0.2 \log 0.2 - 0.2 \log 0.2$$

Multiple Tree - the New Objective Function

- We have an objective function defined on single tree as $\mathcal{G}(\text{tree})$.
- We have introduced entropy of tree[s], noted as $\mathcal{H}(\text{tree[s]})$.

Multiple Tree - the New Objective Function

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

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

- 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.

Building Multiple Trees - Algorithm

Multi-tree Building Algorithm

```
initialize a set of monophone state trees
n = number of leaves per tree that we want
while not all trees have n leaves do
    s = the best split on trees having < n leaves
    apply split s
end while
```

Multi-tree Building Algorithm

```
initialize a set of monophone state trees
n = number of leaves per tree that we want
while not all trees have n leaves do
    s = the best split on trees having < n leaves
    apply split s
end while
while true do
    c = the smallest cost of a merge between any leaves in a same tree
    if c < merge threshold  then
        apply the merge corresponding to c
    else
        terminate and return the trees
    end if
end while
```

Multi-tree Model Training and Decoding

- 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.
    - Merge decoded lattices - *Minimum Bayes Risk* decoding.
    - Our method: to merge acoustic likelihoods.

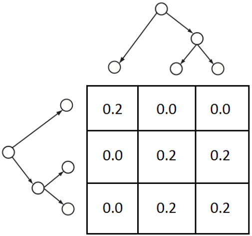
Method to Combine Models

- 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).

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

- Transition probabilities in HMM are relatively much less important and we simply use the arithmetic means of the transition probabilities from each model.

Model Combination Implementation

- 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.
- 

0.2	0.0	0.0
0.0	0.2	0.2
0.0	0.2	0.2

  might have ≥ 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 paramater 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.

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

Impact of the added Entropy Term, cont'ed

- This table shows that the added entropy term does increase the joint-entropy while not increasing single tree entropies too much.

# 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: 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.

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

More results

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

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.