Lecture 7

LVCSR Training and Decoding (Part A)

Bhuvana Ramabhadran, Michael Picheny, Stanley F. Chen

IBM T.J. Watson Research Center
Yorktown Heights, New York, USA
{bhuvana,picheny,stanchen}@us.ibm.com

20 October 2009

The Big Picture

- Weeks 1–4: Small vocabulary ASR.
- Weeks 5–8: Large vocabulary ASR.
  - Week 5: Language modeling (for large vocabularies).
  - Week 6: Pronunciation modeling — acoustic modeling for large vocabularies.
  - Week 7, 8: Training, decoding for large vocabularies.
- Weeks 9–13: Advanced topics.

Outline

- Part I: The LVCSR acoustic model.
- Part II: Acoustic model training for LVCSR.
- Part III: Decoding for LVCSR (inefficient).
  - Part IV: Introduction to finite-state transducers.
- Part V: Search (Lecture 8).
  - Making decoding for LVCSR efficient.

Part I

The LVCSR Acoustic Model

What is LVCSR?

- Large vocabulary.
  - Phone-based modeling *vs.* word-based modeling.
- Continuous.
  - No pauses between words.

The Fundamental Equation of ASR

$$
\begin{aligned}
\text{class}(\mathbf{x}) &= \arg\max_{\omega} P(\omega|\mathbf{x}) \\
&= \arg\max_{\omega} \frac{P(\omega)P(\mathbf{x}|\omega)}{P(\mathbf{x})} \\
&= \arg\max_{\omega} P(\omega)P(\mathbf{x}|\omega)
\end{aligned}
$$

- $P(\mathbf{x}|\omega)$ — acoustic model.
- $P(\omega)$ — language model.

The Acoustic Model: Small Vocabulary

$$
\begin{aligned}
P_\omega(\mathbf{x}) &= \sum_A P_\omega(\mathbf{x}, A) = \sum_A P_\omega(A) \times P_\omega(\mathbf{x}|A) \\
&\approx \max_A P_\omega(A) \times P_\omega(\mathbf{x}|A) \\
&= \max_A \prod_{t=1}^{T} P(a_t) \prod_{t=1}^{T} P(\vec{x}_t|a_t) \\
\log P_\omega(\mathbf{x}) &= \max_A \left[\sum_{t=1}^{T} \log P(a_t) + \sum_{t=1}^{T} \log P(\vec{x}_t|a_t) \right] \\
P(\vec{x}_t|a_t) &= \sum_{m=1}^{M} \lambda_{a_t,m} \prod_{\text{dim } d}^{D} \mathcal{N}(x_{t,d}; \mu_{a_t,m,d}, \sigma_{a_t,m,d})
\end{aligned}
$$

The Acoustic Model: Large Vocabulary

$$
\begin{aligned}
P_\omega(\mathbf{x}) &= \sum_A P_\omega(\mathbf{x}, A) = \sum_A P_\omega(A) \times P_\omega(\mathbf{x}|A) \\
&\approx \max_A P_\omega(A) \times P_\omega(\mathbf{x}|A) \\
&= \max_A \prod_{t=1}^{T} P(a_t) \prod_{t=1}^{T} P(\vec{x}_t|a_t) \\
\log P_\omega(\mathbf{x}) &= \max_A \left[\sum_{t=1}^{T} \log P(a_t) + \sum_{t=1}^{T} \log P(\vec{x}_t|a_t) \right] \\
P(\vec{x}_t|a_t) &= \sum_{m=1}^{M} \lambda_{a_t,m} \prod_{\text{dim } d}^{D} \mathcal{N}(x_{t,d}; \mu_{a_t,m,d}, \sigma_{a_t,m,d})
\end{aligned}
$$

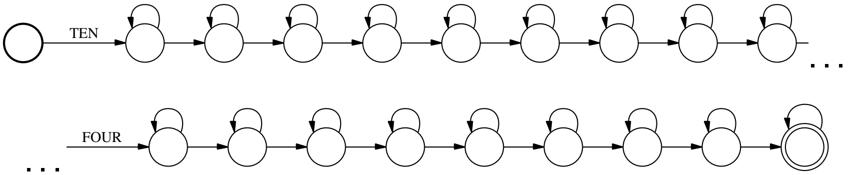
What Has Changed?

- The HMM.
  - Each alignment A describes a path through an HMM.
- Its parameterization.
  - In $P(\vec{x}_t | a_t)$, how many GMM's to use? (Share between HMM's?)

Describing the Underlying HMM

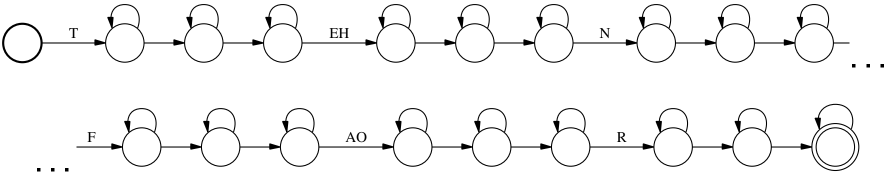
- Fundamental concept: how to map a word (or baseform) sequence to its HMM.
  - In training, map reference transcript to its HMM.
  - In decoding, glue together HMM's for all allowable word sequences.

The HMM: Small Vocabulary



- One HMM per word.
- Glue together HMM for each word in word sequence.

The HMM: Large Vocabulary



- One HMM per phone.
- Glue together HMM for each phone in phone sequence.
  - Map word sequence to phone sequence using baseform dictionary.

I Still Don’t See What’s Changed

- HMM topology typically doesn’t change.
- HMM parameterization changes.

Parameterization

- Small vocabulary.
  - One GMM per state (three states per phone).
  - No sharing between phones in different words.
- Large vocabulary, context-independent (CI).
  - One GMM per state.
  - *Tying* between phones in different words.
- Large vocabulary, context-dependent (CD).
  - Many GMM's per state; GMM to use depends on phonetic context.
  - Tying between phones in different words.

Context-Dependent Parameterization

- Each phone HMM state has its own decision tree.
  - Decision tree asks questions about phonetic context. (Why?)
  - One GMM per leaf in the tree. (Up to 200+ leaves/tree.)
- How will tree for first state of a phone tend to differ . . .
  - From tree for last state of a phone?
- Terminology.
  - *triphone* model — ± 1 phones of context.
  - *quinphone* model — ± 2 phones of context.

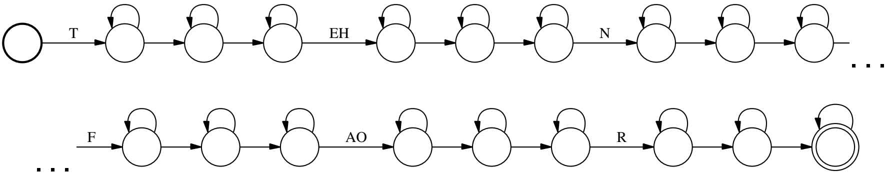
A Real-Life Tree

```
Tree for feneme AA_1:
node   0: quest-P  23[-1] --> true: node   1, false: node   2
  quest: AX AXR B BD CH D DD DH DX D$ ER F G GD HH JH K KD M N NG P PD R S
    SH T TD TH TS UW V W X Z ZH
node   1: quest-P  66[-1] --> true: node   3, false: node   4
  quest: AO AXR ER IY L M N NG OW OY R UH UW W Y
node   2: quest-P  36[-2] --> true: node   5, false: node   6
  quest: D$ X
node   3: quest-P  13[-1] --> true: node   7, false: node   8
  quest: AXR ER R
node   4: quest-P  13[+1] --> true: node   9, false: node  10
  quest: AXR ER R
node   5: leaf    0
node   6: quest-P  15[-1] --> true: node  11, false: node  12
  quest: AXR ER L OW R UW W
node   7: quest-P  49[-2] --> true: node  13, false: node  14
  quest: DX K P T
node   8: quest-P  20[-1] --> true: node  15, false: node  16
  quest: B BD CH D DD DH F G GD IY JH K KD M N NG P PD S SH T TD TH TS V X Y
    Z ZH
node   9: quest-P  43[-2] --> true: node  17, false: node  18
  quest: CH DH F HH JH S SH TH TS V Z ZH
node  10: quest-P  49[-1] --> true: node  19, false: node  20
  quest: DX K P T
node  11: leaf    1
node  12: quest-P  15[-2] --> true: node  21, false: node  22
  quest: AXR ER L OW R UW W
node  13: leaf    2
node  14: leaf    3
...
```

Pop Quiz

- Pretend you are Keanu Reeves.
- System description:
  - 1000 words in lexicon; average word length = 5 phones.
  - There are 50 phones; each phone HMM has three states.
  - Each decision tree contains 100 leaves on average.
- How many GMM's are there in:
  - A small vocabulary system (word models)?
  - A CI large vocabulary system?
  - A CD large vocabulary system?

Any Questions?



- Given a word sequence, you should understand how to ...
  - Layout the corresponding HMM topology.
  - Determine which GMM to use at each state, for CI and CD models.

Context-Dependent Phone Models

- Typical model sizes:

type	HMM	GMM's/ state	GMM's	Gaussians
word	per word	1	10–500	100–10k
CI phone	per phone	1	$\sim$150	1k–3k
CD phone	per phone	1–200	1k–10k	10k–300k

- 39-dimensional feature vectors $\Rightarrow$ $\sim$80 parameters/Gaussian.
- Big models can have tens of millions of parameters.

What About Transition Probabilities?

- This slide only included for completeness.
- Small vocabulary.
  - One set of transition probabilities per state.
  - No sharing between phones in different words.
- Large vocabulary.
  - One set of transition probabilities per state.
  - Sharing between phones in different words.
- What about context-dependent transition modeling?

Recap

- Main difference between small vocabulary and large vocabulary:
  - Allocation of GMM's.
  - Sharing GMM's between words: needs less GMM's.
  - Modeling context-dependence: needs more GMM's.
  - Hybrid allocation is possible.
- Training and decoding for LVCSR.
  - In theory, any reason why small vocabulary techniques won't work?
  - In practice, yikes!

Points to Ponder

- Why deterministic mapping?
  - DID YOU $\Rightarrow$ D IH D JH UW
  - The area of *pronunciation modeling*.
- Why decision trees?
  - Unsupervised clustering.

Part II

Acoustic Model Training for LVCSR

Small Vocabulary Training — Lab 2

- Phase 1: Collect underpants.
  - Initialize all Gaussian means to 0, variances to 1.
- Phase 2: Iterate over training data.
  - For each word, train associated word HMM . . .
  - On all samples of that word in the training data . . .
  - Using the Forward-Backward algorithm.
- Phase 3: Profit!

Large Vocabulary Training

- What's changed going to LVCSR?
  - Same HMM topology; just more Gaussians and GMM's.
- Can we just use the same training algorithm as before?

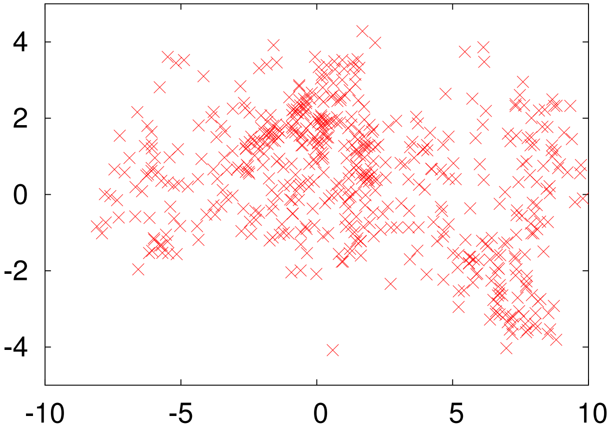
Where Are We?

Flat or Random Start

- Why does this work for small models?
  - We believe there's a huge global minimum . . .
  - In the "middle" of the parameter search space.
  - With a neutral starting point, we're apt to fall into it.
  - (Who knows if this is actually true.)
- Why doesn't this work for large models?

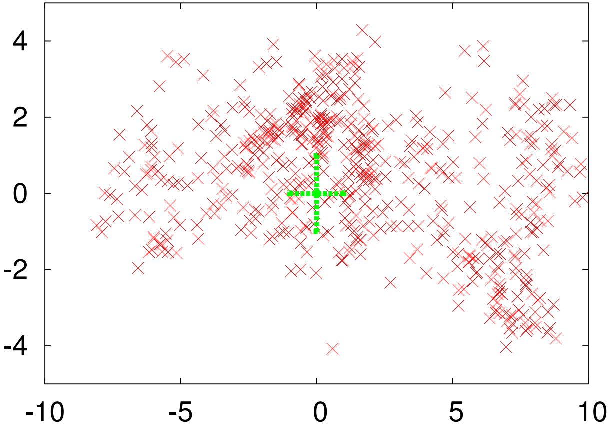
Case Study: Training a Simple GMM

- Front end from Lab 1; first two dimensions; 546 frames.



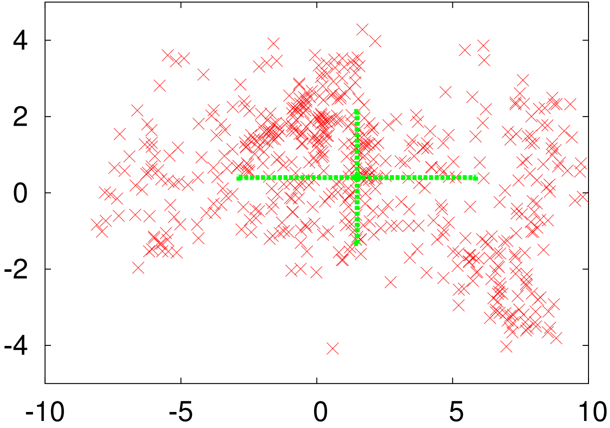
Training a Mixture of Two 2-D Gaussians

- Flat start?
  - Initialize mean of each Gaussian to 0, variance to 1.



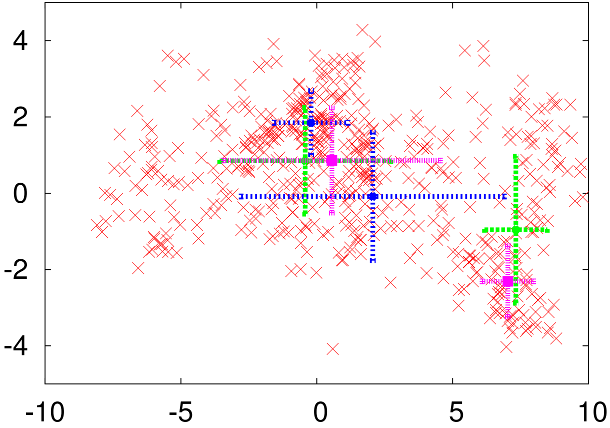
Training a Mixture of Two 2-D Gaussians

- "*At the Mr. O level, symmetry is everything.*"
  - At the GMM level, symmetry is a bad idea.



Training a Mixture of Two 2-D Gaussians

- Random seeding?
  - Picked 8 random starting points ⇒ 3 different optima.



Training Hidden Models

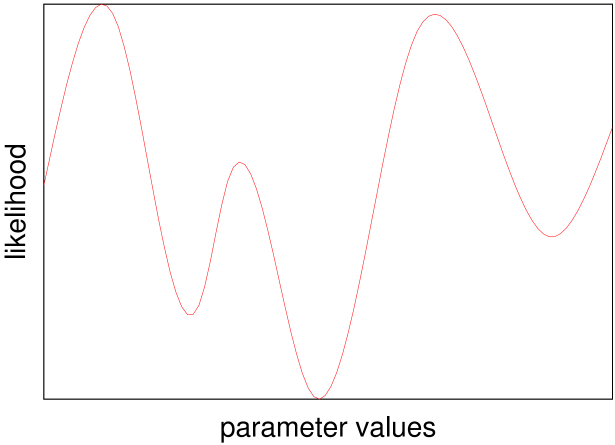
- (MLE) training of models with hidden variables has local minima.
- What are the hidden variables in ASR?
  - *i.e.*, what variables are in our model . . .
  - That are not observed.

How To Spot Hidden Variables

$$
\begin{aligned}
P_\omega(\mathbf{x}) &= \sum_A P_\omega(\mathbf{x}, A) = \sum\nolimits_A P_\omega(A) \times P_\omega(\mathbf{x}|A) \\
&\approx \max_A P_\omega(A) \times P_\omega(\mathbf{x}|A) \\
&= \max_A \prod_{t=1}^{T} P(a_t) \prod_{t=1}^{T} P(\vec{x}_t|a_t) \\
\log P_\omega(\mathbf{x}) &= \max_A \left[\sum_{t=1}^{T} \log P(a_t) + \sum_{t=1}^{T} \log P(\vec{x}_t|a_t) \right] \\
P(\vec{x}_t|a_t) &= \sum\nolimits_{m=1}^{M} \lambda_{a_t,m} \prod_{\text{dim } d}^{D} \mathcal{N}(x_{t,d}; \mu_{a_t,m,d}, \sigma_{a_t,m,d})
\end{aligned}
$$

Gradient Descent and Local Minima

- EM training does hill-climbing/gradient descent.
  - Finds "nearest" optimum to where you started.



What To Do?

- Insight: If we know the “correct” hidden values for a model:
  - *e.g.*, which arc and which Gaussian for each frame . . .
  - Training is easy! (No local minima.)
  - Remember Viterbi training given fixed alignment in Lab 2.
- Is there a way to guess the correct hidden values for a large model?

Bootstrapping Alignments

- Recall that all of our acoustic models, from simple to complex:
  - Generally use the same HMM topology!
  - (All that differs is how we assign GMM's to each arc.)
- Given an alignment (from arc/phone states to frames) for simple model . . .
  - It is straightforward to compute analogous alignment for complex model!

Bootstrapping Big Models From Small

- Recipe:
  - Start with model simple enough that flat start works.
  - Iteratively build more and more complex models . . .
  - By using last model to seed hidden values for next.
- Need to come up with sequence of successively more complex models . . .
  - With related hidden structure.

How To Seed Next Model From Last

- Directly via hidden values, *e.g.*, alignment.
  - *e.g.*, *single-pass retraining*.
  - Can be used between very different models.
- Via parameters.
  - Seed parameters in complex model so that . . .
  - Initially, will yield same/similar alignment as in simple model.
  - *e.g.*, moving from CI to CD GMM's.

Bootstrapping Big Models From Small

- Recurring motif in acoustic model training.
- The reason why state-of-the-art systems ...
  - Require many, many training passes, as you will see.
- Recipes handed down through the generations.
  - Discovered via sweat and tears.
  - Art, not science.
  - But no one believes these find global optima ...
  - Even for small problems.

Overview of Training Process

- Build CI single Gaussian model from flat start.
- Use CI single Gaussian model to seed CI GMM model.
- Build phonetic decision tree (using CI GMM model to help).
- Use CI GMM model to seed CD GMM model.

Where Are We?

Case Study: Training a GMM

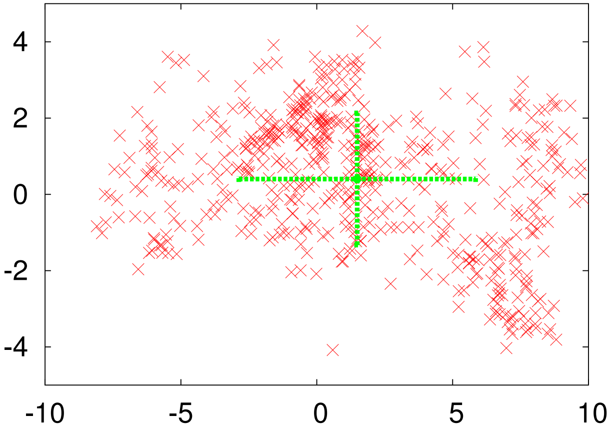
- Recursive mixture splitting.
  - A sequence of successively more complex models.
- *k*-means clustering.
  - Seed means in one shot.

Gaussian Mixture Splitting

- Start with single Gaussian per mixture (trained).
- Split each Gaussian into two.
  - Perturb means in opposite directions; same variance.
  - Train.
- Repeat until reach desired number of mixture components (1, 2, 4, 8, ...).
  - (Discard Gaussians with insufficient counts.)
- Assumption: c-component GMM gives good guidance ...
  - On how to seed $2c$-component GMM.

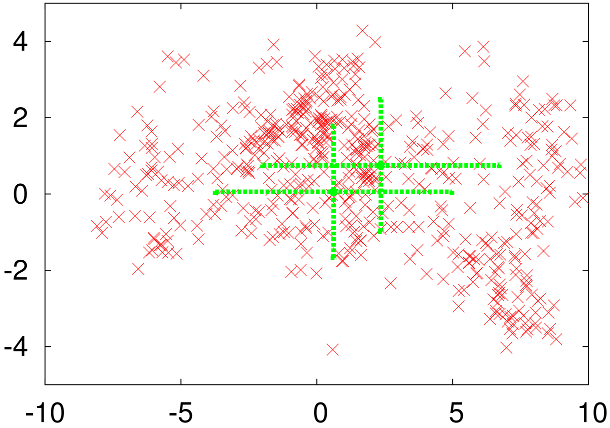
Mixture Splitting Example

- Train single Gaussian.



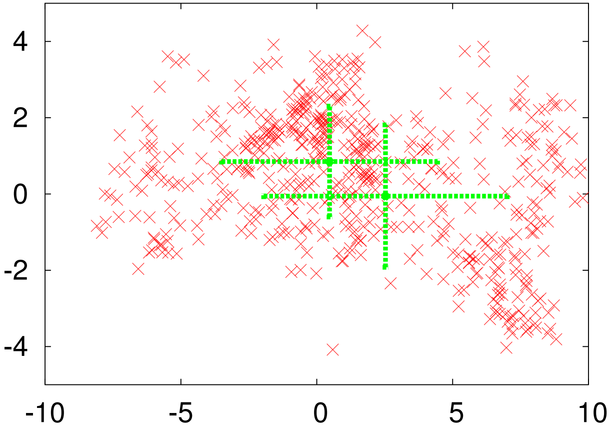
Mixture Splitting Example

- Split each Gaussian in two ($\pm 0.2 \times \vec{\sigma}$)



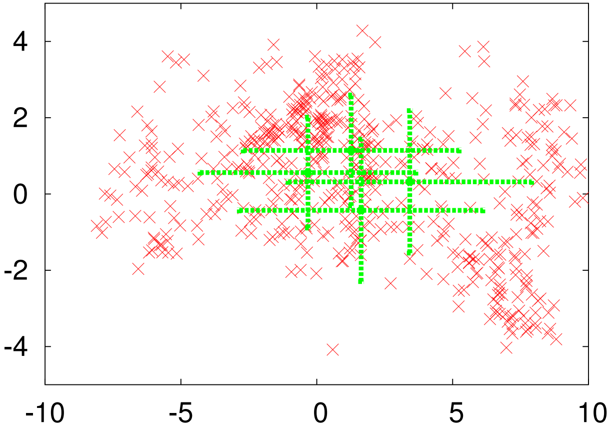
Mixture Splitting Example

- Train, yep.



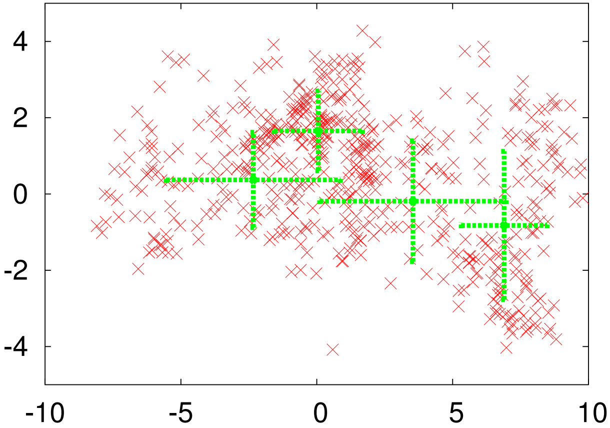
Mixture Splitting Example

- Split each Gaussian in two ($\pm 0.2 \times \vec{\sigma}$)



Mixture Splitting Example

- Train, yep.



Applying Mixture Splitting in ASR

- Recipe:
  - Start with model with 1-component GMM's (à la Lab 2).
  - Split Gaussians in each output distribution simultaneously.
  - Do many iterations of FB.
  - Repeat.
- Real-life numbers:
  - Five splits spread within 30 iterations of FB.

Another Way: Automatic Clustering

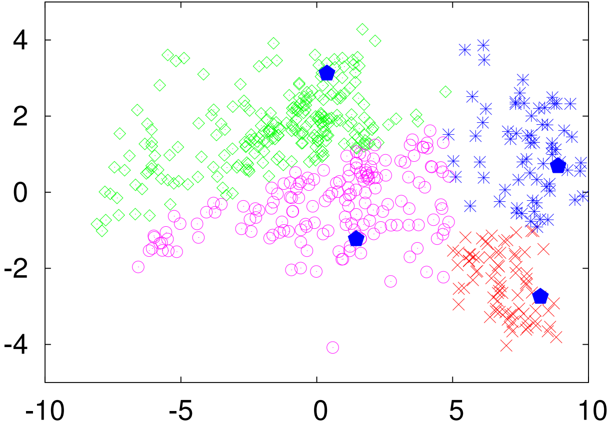
- Use unsupervised clustering algorithm to find clusters.
- Given clusters . . .
  - Use cluster centers to seed Gaussian means.
  - FB training.
  - (Discard Gaussians with insufficient counts.)

k-Means Clustering

- Select desired number of clusters k.
- Choose k data points randomly.
  - Use these as initial cluster centers.
- "Assign" each data point to nearest cluster center.
- Recompute each cluster center as . . .
  - Mean of data points "assigned" to it.
- Repeat until convergence.

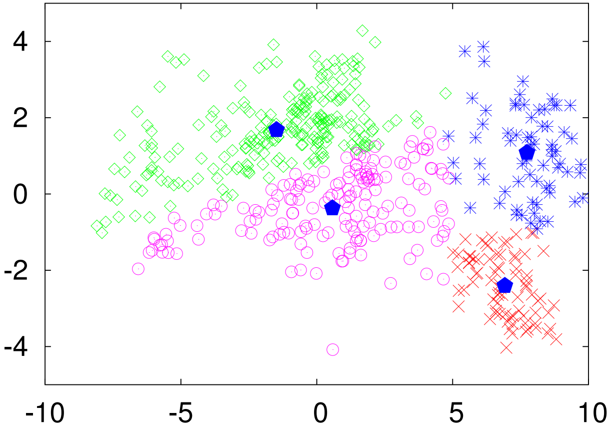
k-Means Example

- Pick random cluster centers; assign points to nearest center.



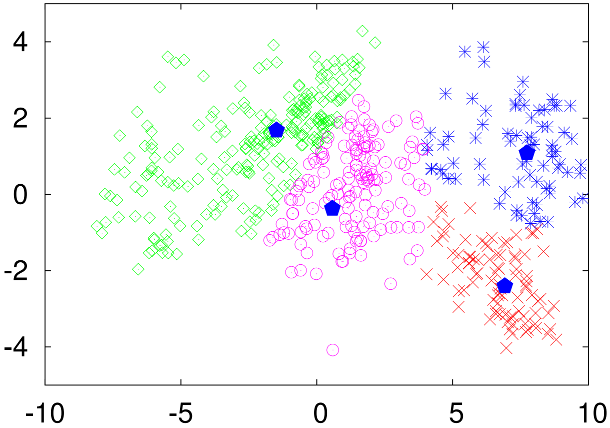
k-Means Example

- Recompute cluster centers.



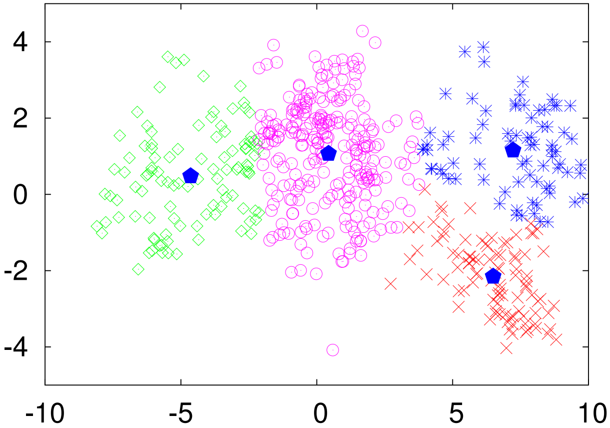
k-Means Example

- Assign each point to nearest center.



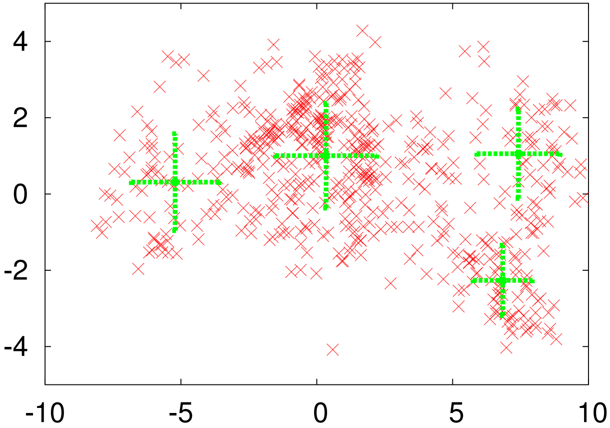
k-Means Example

- Repeat until convergence.

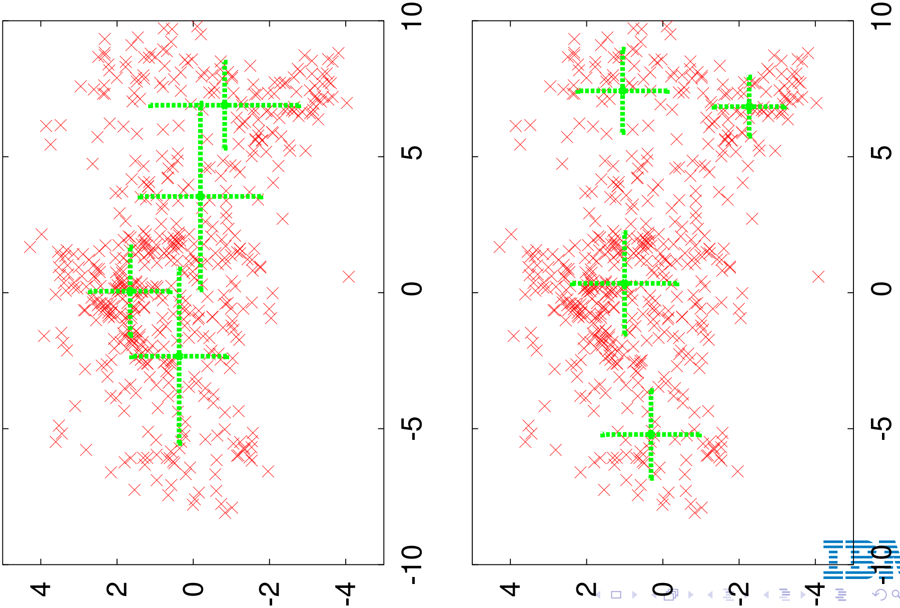


k-Means Example

- Use centers as means of Gaussians; train, yep.



The Final Mixtures, Splitting *vs.* *k*-Means



Technical Aside: *k*-Means Clustering

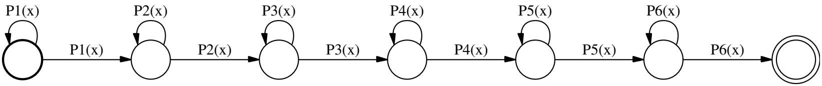
- When using Euclidean distance . . .
- *k*-means clustering is equivalent to . . .
  - Seeding Gaussian means with the *k* initial centers.
  - Doing Viterbi EM update, keeping variances constant.

Applying k-Means Clustering in ASR

- To train each GMM, use k-means clustering ...
  - On what data? Which frames?
- Huh?
  - How to decide which frames *align* to each GMM?
- This issue is evaded for mixture splitting.
  - Can we avoid it here?

Forced Alignment

- Viterbi algorithm.
    - Finds most likely alignment of HMM to data.



frame	0	1	2	3	4	5	6	7	8	9	10	11	12
arc	P_1	P_1	P_1	P_2	P_3	P_4	P_4	P_5	P_5	P_5	P_5	P_6	P_6

- Need existing model to create alignment. (Which?)

Recap

- You can use single Gaussian models to seed GMM models.
  - Mixture splitting: use c-component GMM to seed $2c$-component GMM.
  - k-means: use single Gaussian model to find alignment.
- Both of these techniques work about the same.
  - Nowadays, we primarily use mixture splitting.

Where Are We?

What Do We Need?

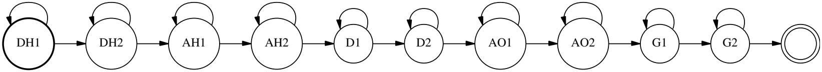
- For each tree/phone state ...
  - List of frames/feature vectors associated with that tree.
  - (This is the data we are optimizing the likelihood of.)
  - For each frame, the phonetic context.
- A list of candidate questions about the phonetic context.
  - Ask about phonetic concepts; *e.g.*, vowel or consonant?
  - Expressed as list of phones in set.
  - Allow same questions to be asked about each phone position.
  - Handed down through the generations.

A Real-Life Tree

```
Tree for feneme AA_1:
node   0: quest-P  23[-1] --> true: node   1, false: node   2
  quest: AX AXR B BD CH D DD DH DX D$ ER F G GD HH JH K KD M N NG P PD R S
    SH T TD TH TS UW V W X Z ZH
node   1: quest-P  66[-1] --> true: node   3, false: node   4
  quest: AO AXR ER IY L M N NG OW OY R UH UW W Y
node   2: quest-P  36[-2] --> true: node   5, false: node   6
  quest: D$ X
node   3: quest-P  13[-1] --> true: node   7, false: node   8
  quest: AXR ER R
node   4: quest-P  13[+1] --> true: node   9, false: node  10
  quest: AXR ER R
node   5: leaf     0
node   6: quest-P  15[-1] --> true: node  11, false: node  12
  quest: AXR ER L OW R UW W
node   7: quest-P  49[-2] --> true: node  13, false: node  14
  quest: DX K P T
node   8: quest-P  20[-1] --> true: node  15, false: node  16
  quest: B BD CH D DD DH F G GD IY JH K KD M N NG P PD S SH T TD TH TS V X Y
    Z ZH
node   9: quest-P  43[-2] --> true: node  17, false: node  18
  quest: CH DH F HH JH S SH TH TS V Z ZH
node  10: quest-P  49[-1] --> true: node  19, false: node  20
  quest: DX K P T
node  11: leaf     1
node  12: quest-P  15[-2] --> true: node  21, false: node  22
  quest: AXR ER L OW R UW W
node  13: leaf     2
node  14: leaf     3
...
```

Training Data for Decision Trees

- Forced alignment/Viterbi decoding!
- Where do we get the model to align with?
  - Use CI phone model or other pre-existing model.



frame	0	1	2	3	4	5	6	7	8	9	...
arc	DH_1	DH_2	AH_1	AH_2	D_1	D_1	D_2	D_2	D_2	AO_1	...

Building the Tree

- A set of events $\{(\vec{x}_i, p_L, p_R)\}$ (possibly subsampled).
- Given current tree:
  - Choose question of the form . . .
  - "*Does the phone in position j belong to the set q?*" . . .
  - That optimizes $\prod_i P(\vec{x}_i|\text{leaf}(p_L, p_R))$ . . .
  - Where we model each leaf using a single Gaussian.
- Can efficiently build whole level of tree in single pass.
- See Lecture 6 slides and readings for the gory details.

Seeding the Context-Dependent GMM's

- Context-independent GMM's: one GMM per phone state.
- Context-dependent GMM's: l GMM's per phone state.
- How to seed context-dependent GMM's?
  - *e.g.*, so that initial alignment matches CI alignment?

Where Are We?

Where Are We?

4 Details
- Maximum Likelihood Training?
- Viterbi *vs.* Non-Viterbi Training
- Graph Building

The Original Story, Small Vocabulary

- One HMM for each word; flat start.
- Collect all examples of each word.
  - Run FB on those examples to do maximum likelihood training of that HMM.

The New Story

- One HMM for each word sequence!?
  - But tie parameters across HMM's!
- Do complex multi-phase training.
- Are we still doing anything resembling maximum likelihood training?

Maximum Likelihood Training?

- Regular training iterations (FB, Viterbi EM).
  - Increase (Viterbi) likelihood of data.
- Seeding last model from next model.
  - Mixture splitting.
  - CI $\Rightarrow$ CD models.
- (Decision-tree building.)

Maximum Likelihood Training?

- Just as LM's need to be smoothed or *regularized*.
  - So do acoustic models.
  - Prevent extreme likelihood values (*e.g.*, 0 or ∞).
- ML training maximizes training data likelihood.
  - We actually want to optimize test data likelihood.
  - Let's call the difference the *overfitting penalty*.
- The overfitting penalty tends to increase as . . .
  - The number of parameters increase and/or . . .
  - Parameter magnitudes increase.

Regularization/Capacity Control

- Limit size of model.
  - Will training likelihood continue to increase as model grows?
  - Limit components per GMM.
  - Limit number of leaves in decision tree, *i.e.*, number of GMM's.
- Variance flooring.
  - Don't let variances go to 0 $\Rightarrow$ infinite likelihood.

Where Are We?

4 Details

- Maximum Likelihood Training?
- Viterbi *vs.* Non-Viterbi Training
- Graph Building

Two Types of Updates

- "Full" EM.
  - Compute true posterior of each hidden configuration.
- Viterbi EM.
  - Use Viterbi algorithm to find most likely hidden configuration.
  - Assign posterior of 1 to this configuration.
- Both are valid updates; instances of generalized EM.

Examples

- Training GMM's.
  - Mixture splitting *vs.* k-means clustering.
- Training HMM's.
  - Forward-backward *vs.* Viterbi EM (Lab 2).
- Everywhere you do a forced alignment.
  - Refining the reference transcript.
  - What is non-Viterbi version of decision-tree building?

When To Use One or the Other?

- Which version is more expensive computationally?
  - Optimization: need not realign every iteration.
- Which version finds better minima?
- If posteriors are very sharp, they do almost the same thing.
  - Remember example posteriors in Lab 2?
- Rule of thumb:
  - When you're first training a "new" model, use full EM.
  - Once you're "locked in" to an optimum, Viterbi is fine.

Where Are We?

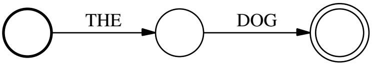
4 Details
- Maximum Likelihood Training?
- Viterbi *vs.* Non-Viterbi Training
- Graph Building

Building HMM's For Training

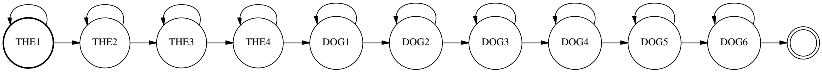
- When doing Forward-Backward on an utterance ...
  - We need the HMM corresponding to the reference transcript.
- Can we use the same techniques as for small vocabularies?

Word Models

- Reference transcript

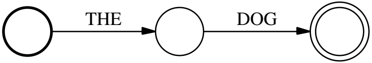


- Replace each word with its HMM

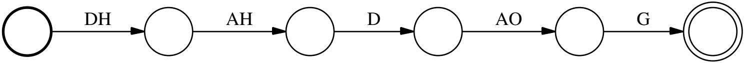


Context-Independent Phone Models

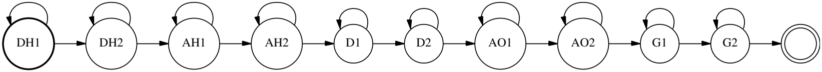
- Reference transcript



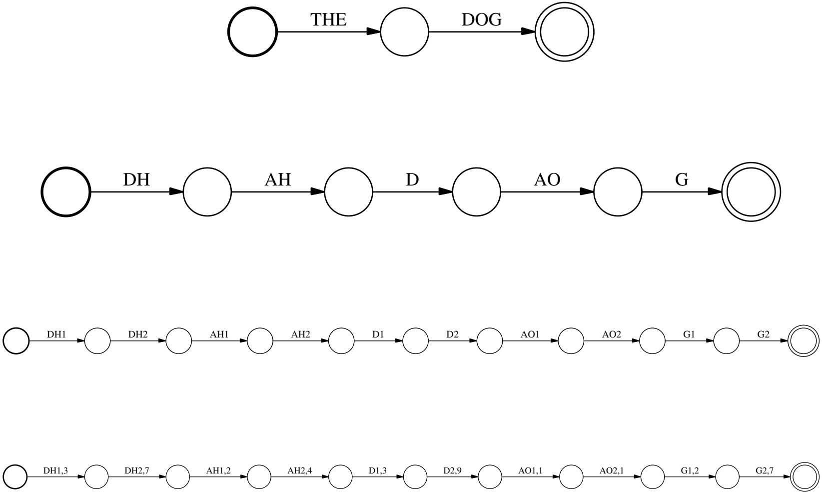
- Pronunciation dictionary.
    - Maps each word to a sequence of phonemes.



- Replace each phone with its HMM



Context-Dependent Phone Models



The Pronunciation Dictionary

- Need pronunciation of *every* word in training data.
  - Including pronunciation variants

    ```
    THE(01)    DH  AH
    THE(02)    DH  IY
    ```

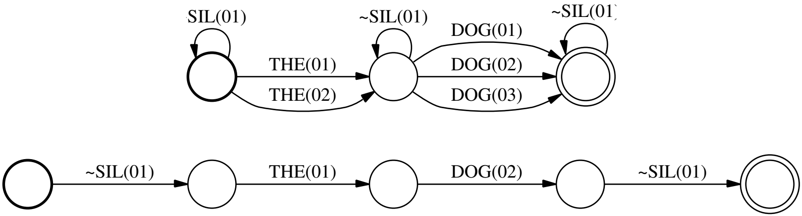
  - Listen to data?
  - Use automatic spelling-to-sound models?
- Why not consider multiple baseforms/word for word models?

But Wait, It's More Complicated Than That!

- Reference transcripts are created by humans . . .
  - Who, by their nature, are *human* (*i.e.*, fallible)
- Typical transcripts don't contain everything an ASR system wants.
  - Where silence occurred; noises like coughs, door slams, etc.
  - Pronunciation information, *e.g.*, was `THE` pronounced as `DH UH` or `DH IY`?

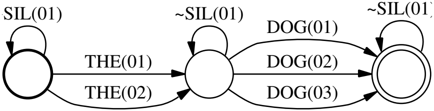
Pronunciation Variants, Silence, and Stuff

- How can we produce a more "complete" reference transcript?
- Viterbi decoding!
  - Build HMM accepting all word (HMM) sequences consistent with reference transcript.
  - Compute best path/word HMM sequence.
  - Where does this initial acoustic model come from?



Another Way

- Just use the whole expanded graph during training.



- The problem: how to do context-dependent phone expansion?
    - Use same techniques as in building graphs for decoding.

Where Are We?

Prerequisites

- Audio data with reference transcripts.
- What two other things?

The Training Recipe

- Find/make baseforms for all words in reference transcripts.
- Train single Gaussian models (flat start; many iters of FB).
- Do mixture splitting, say.
  - Split each Gaussian in two; do many iterations of FB.
  - Repeat until desired number of Gaussians per mixture.
- (Use initial system to refine reference transcripts.)
  - Select pronunciation variants, where silence occurs.
  - Do more FB training given refined transcripts.
- Build phonetic decision tree.
  - Use CI model to align training data.
- Seed CD model from CI; train using FB or Viterbi EM.
  - Possibly doing more mixture splitting.

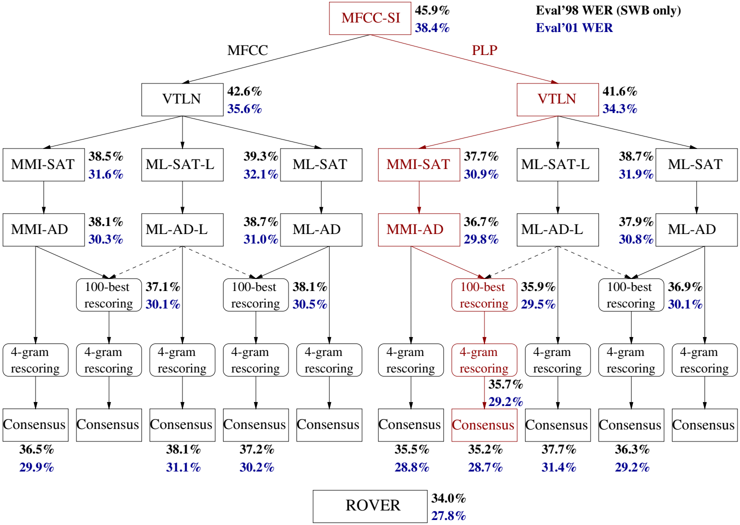
How Long Does Training Take?

- It's a secret.
- We think in terms of *real-time factor*.
  - How many hours does it take to process one hour of speech?

Whew, That Was Pretty Complicated!

- Adaptation (VTLN, fMLLR, mMLLR)
- Discriminative training (LDA, MMI, MPE, fMPE)
- Model combination (cross adaptation, ROVER)
- Iteration.
  - Repeat steps using better model for seeding.
  - Alignment is only as good as model that created it.

Things Can Get Pretty Hairy



Recap: Acoustic Model Training for LVCSR

- Take-home messages.
  - Hidden model training is fraught with local minima.
  - Seeding more complex models with simpler models helps avoid terrible local minima.
  - People have developed many recipes/heuristics to try to improve the minimum you end up in.
  - Training is insanely complicated for state-of-the-art research models.
- The good news . . .
  - I just saved a bunch on money on my car insurance by switching to GEICO.

Part III

Decoding for LVCSR (Inefficient)

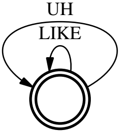
Decoding for LVCSR (Inefficient)

$$\begin{aligned}\text{class}(\mathbf{x}) &= \arg\max_{\omega} P(\omega|\mathbf{x}) \\ &= \arg\max_{\omega} \frac{P(\omega)P(\mathbf{x}|\omega)}{P(\mathbf{x})} \\ &= \arg\max_{\omega} P(\omega)P(\mathbf{x}|\omega)\end{aligned}$$

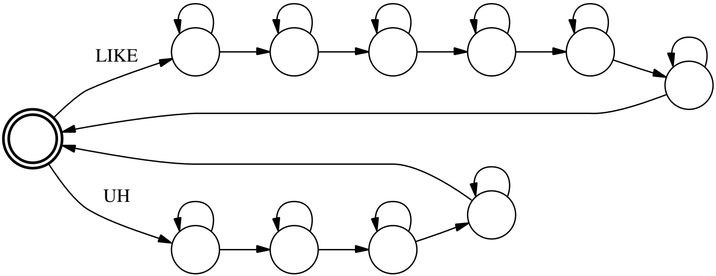
- Now that we know how to build models for LVCSR …
  - CD acoustic models via complex recipes.
  - *n*-gram models via counting and smoothing.
- How can we use them for decoding?
  - Let's ignore memory and speed constraints for now.

Decoding: Small Vocabulary

- Take graph/WFSA representing language model
  - *i.e.*, all allowable word sequences.



- Expand to underlying HMM

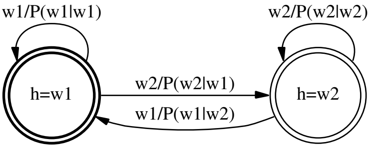


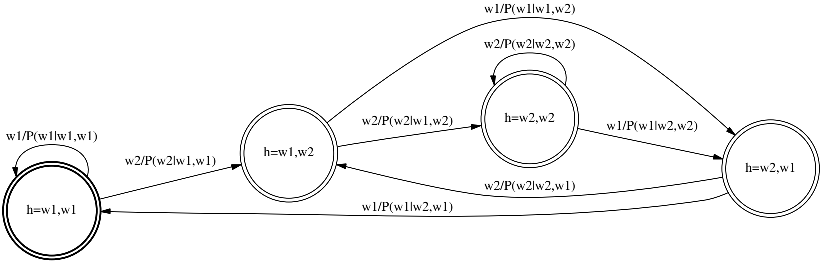
- Run the Viterbi algorithm!

Issue 1: Are N-Gram Models WFSA's?

- Yup.
- Invariants.
  - One state for each $(n-1)$-gram history.
  - All paths ending in state for $(n-1)$-gram ω ...
  - Are labeled with word sequence ending in ω.
  - State for $(n-1)$-gram ω has outgoing arc for each word w ...
  - With arc probability $P(w|\omega)$.

Bigram, Trigram LM's Over Two Word Vocab



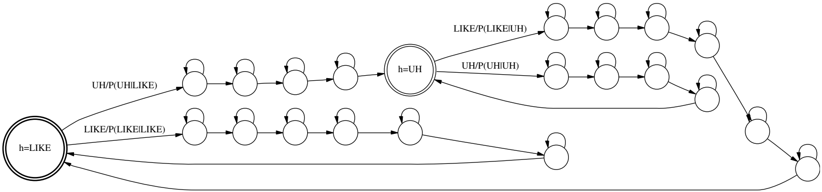


Pop Quiz

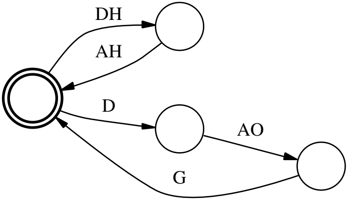
- How many states in FSA representing n-gram model ...
  - With vocabulary size $|V|$?
- How many arcs?

Issue 2: Graph Expansion

- Word models.
  - Replace each word with its HMM.
- CI phone models.
  - Replace each word with its phone sequence(s)
  - Replace each phone with its HMM.

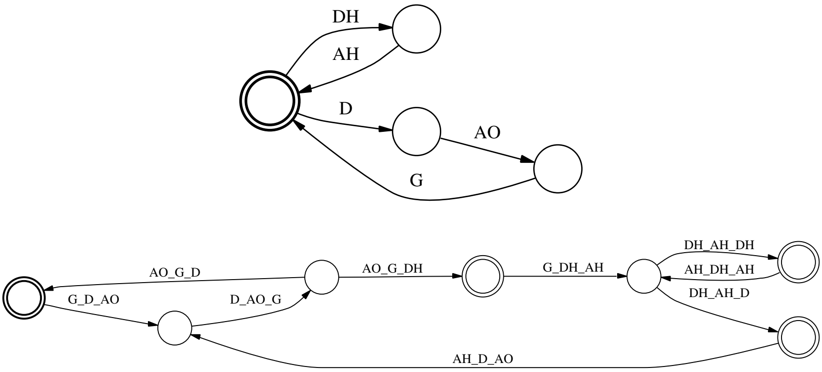


Context-Dependent Graph Expansion



- How can we do context-dependent expansion?
  - Handling branch points is tricky.
- Other tricky cases.
  - Words consisting of a single phone.
  - Quinphone models.

Triphone Graph Expansion Example



Word-Internal Acoustic Models

- Simplify acoustic model to simplify graph expansion.
- *Word-internal* models.
  - Don't let decision trees ask questions across word boundaries.
  - Pad contexts with the *unknown phone*.
  - Hurts performance (*e.g.*, coarticulation across words).
- As with word models, just replace each word with its HMM.

Context-Dependent Graph Expansion

- Is there some elegant theoretical framework . . .
- That makes it easy to do this type of expansion . . .
- And also makes it easy to do lots of other graph operations useful in ASR?
- $\Rightarrow$ Finite-state transducers (FST's)! (Part IV)

Recap: Decoding for LVCSR (Inefficient)

- In theory, do same thing as we did for small vocabularies.
  - Start with LM represented as word graph.
  - Expand to underlying HMM.
  - Viterbi.
- In practice, computation and memory issues abound.
- How to do the graph expansion? FST's (Part IV)
- How to make decoding efficient? search (Part V)

Part IV

Introduction to Finite-State Transducers

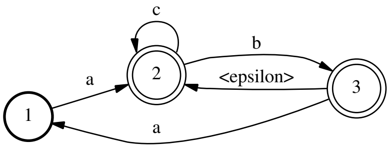
Introduction to Finite-State Transducers

Overview

- FST's are closely related to finite-state automata (FSA).
  - An FSA is a graph.
  - An FST ...
  - Takes an FSA as input ...
  - And produces a new FSA.
- Natural technology for graph expansion ...
  - And much, much more.
- FST's for ASR pioneered by AT&T in late 1990's

Review: What is a Finite-State Acceptor?

- It has states.
  - Exactly one initial state; one or more final states.
- It has arcs.
  - Each arc has a label, which may be empty (ϵ).
- Ignore probabilities for now.
- Meaning: a (possibly infinite) list of strings.

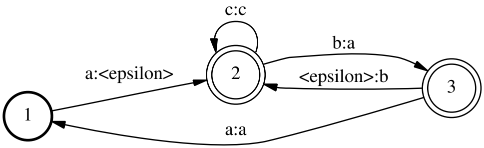


Review: Pop Quiz

- What are the differences between the following:
  - HMM's with discrete output distributions.
  - FSA's with arc probabilities.

What is a Finite-State Transducer?

- It's like a finite-state acceptor, except . . .
- Each arc has two labels instead of one.
  - An *input* label (possibly empty)
  - An *output* label (possibly empty)
- Meaning: a (possibly infinite) list of pairs of strings . . .
  - An input string and an output string.



Terminology

- *finite-state acceptor* (FSA): one label on each arc.
- *finite-state transducer* (FST): input and output label on each arc.
- *finite-state machine* (FSM): FSA or FST.
  - Also, *finite-state automaton*
- Incidentally, an FSA can act like an FST.
  - Pretend input label is both input and output label.

Transforming a Single String

- Let's say you have a string, *e.g.*,

  THE DOG

- Let's say we want to apply a transformation.
  - *e.g.*, map words to their baseforms.

  DH AH D AO G

- This is easy, *e.g.*, use `sed` or `perl` or ...

Transforming Lots of Strings At Once

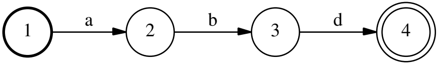
- Let's say you have a (possibly infinite) list of strings ...
  - Expressed as an FSA, as this is compact.
- Let's say we want to apply a transformation.
  - *e.g.*, map words to their baseforms.
- On all of these strings.
- And have the (possibly infinite) list of output strings ...
  - Expressed as an FSA, as this is compact.
- Efficiently.

The Composition Operation

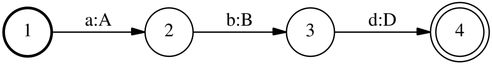
- FSA: represents a list of strings $\{i_1 \cdots i_N\}$.
- FST: represents a list of strings pairs $\{(i_1 \cdots i_N, o_1 \cdots o_M)\}$.
  - A compact way of representing string transformations.
- Composing FSA A with FST T to get FSA $A \circ T$.
  - If string $i_1 \cdots i_N \in A$ and ...
  - Input/output string pair $(i_1 \cdots i_N, o_1 \cdots o_M) \in T$, ...
  - Then, string $o_1 \cdots o_M \in A \circ T$.

Rewriting a Single String

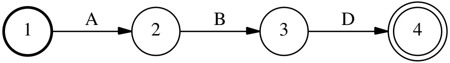
A



T

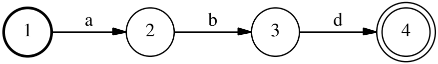


$A \circ T$

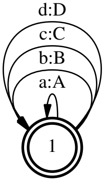


Rewriting a Single String

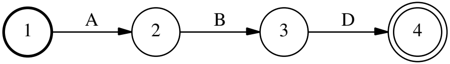
A



T

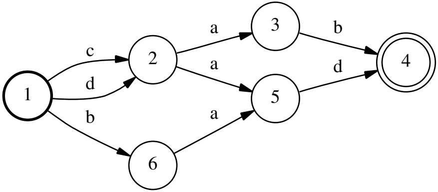


$A \circ T$

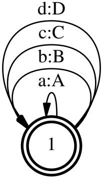


Rewriting Many Strings At Once

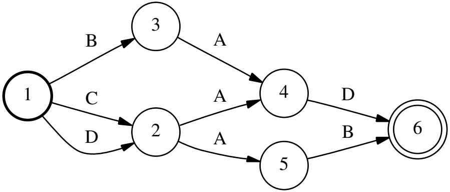
A



T

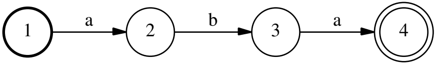


$A \circ T$

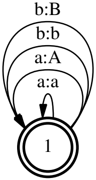


Rewriting A Single String Many Ways

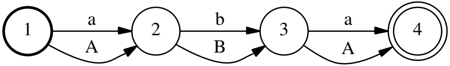
A



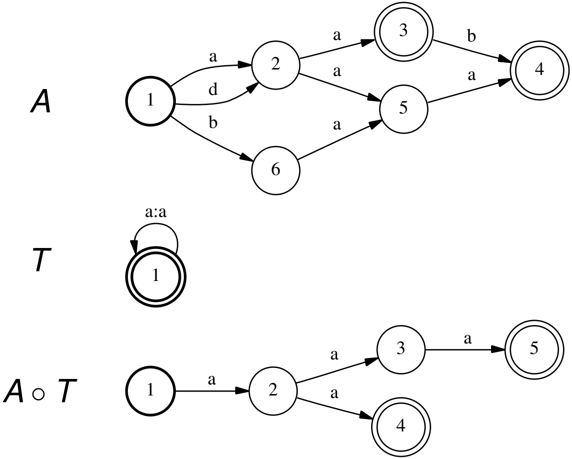
T



$A \circ T$



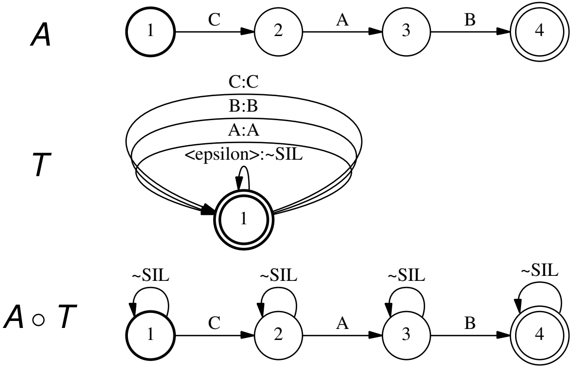
Rewriting Some Strings Zero Ways



And a Dessert Topping!

- Composition seems pretty versatile.
- Can it help us build decoding graphs?

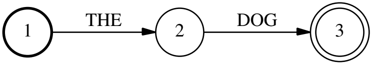
Example: Inserting Optional Silences



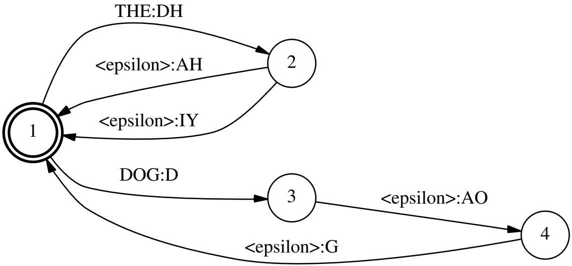
Example: Mapping Words To Phones

THE(01)	DH	AH
THE(02)	DH	IY

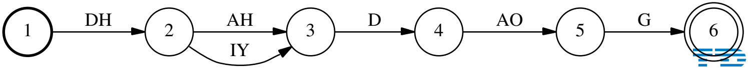
A



T

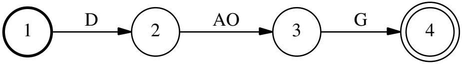


$A \circ T$

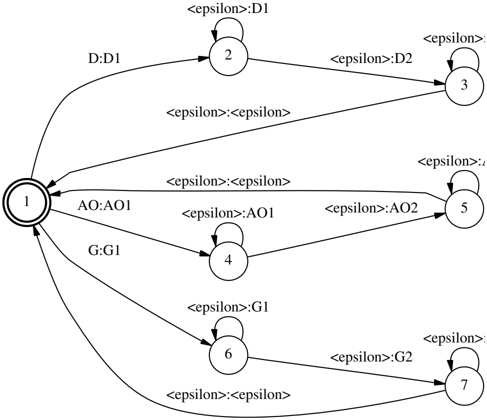


Example: Rewriting CI Phones as HMM's

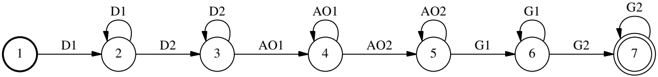
A



T



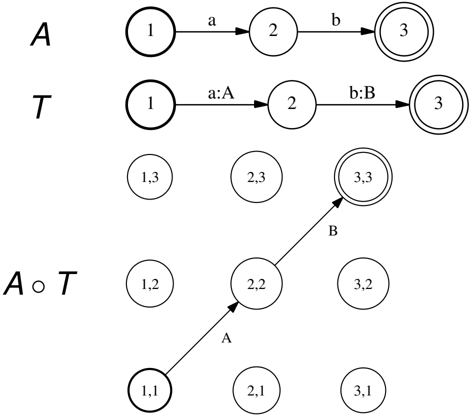
$A \circ T$



Computing Composition

- For now, pretend no ϵ-labels
- For every state $s \in A$, $t \in T$, create state $(s, t) \in A \circ T$
- Create arc from (s_1, t_1) to (s_2, t_2) with label o iff ...
  - There is an arc from s_1 to s_2 in A with label i
  - There is an arc from t_1 to t_2 in T with input label i and output label o
- (s, t) is initial iff s and t are initial; similarly for final states.
- (Remove arcs and states that cannot reach both an initial and final state.)
- What is time complexity?

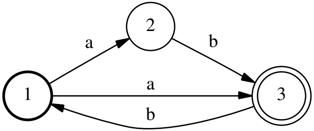
Example: Computing Composition



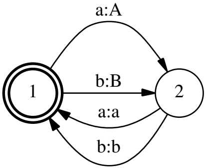
- Optimization: start from initial state, build outward.

Another Example

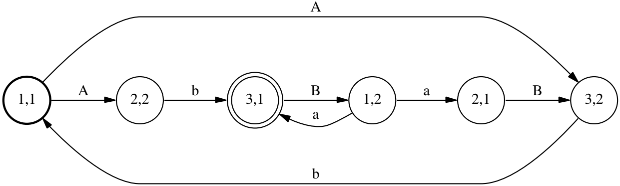
A



T



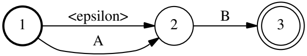
$A \circ T$

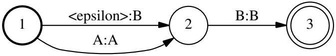


Composition and ϵ-Transitions

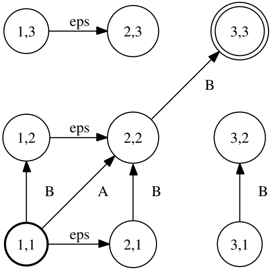
- Basic idea: can take ϵ-transition in one FSM without moving in other FSM.
  - A little tricky to do exactly right.
  - Do the readings if you care: (Pereira, Riley, 1997)

A, T





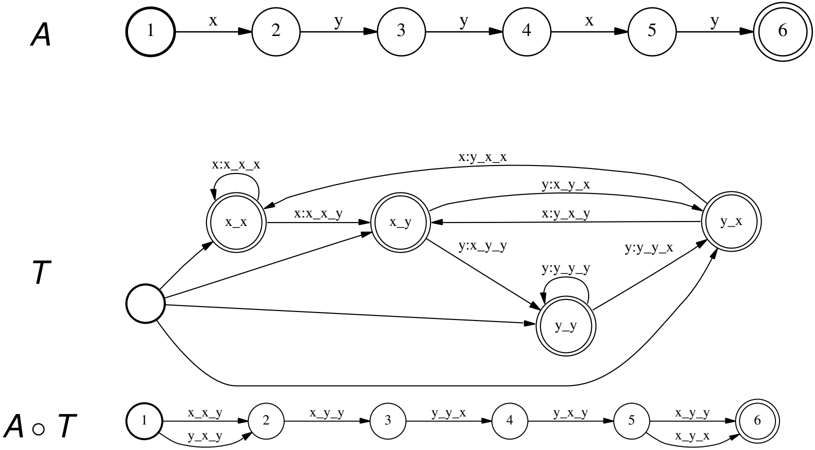
$A \circ T$



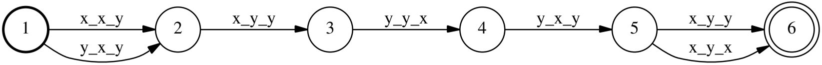
How to Express CD Expansion via FST's?

- Step 1: Rewrite each phone as a triphone.
  - Rewrite `AX` as `DH_AX_R` if `DH` to left, `R` to right.
- Step 2: Rewrite each triphone with correct context-dependent HMM for center phone.
  - Just like rewriting a CI phone as its HMM.
  - Need to precompute HMM for each possible triphone ($\sim 50^3$).

How to Express CD Expansion via FST's?



How to Express CD Expansion via FST's?



- Point: composition automatically expands FSA to correctly handle context!
  - Makes multiple copies of states in original FSA . . .
  - That can exist in different triphone contexts.
  - (And makes multiple copies of *only* these states.)

Recap: Finite-State Transducers

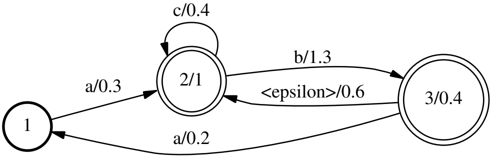
- Graph expansion can be expressed as series of composition operations.
  - Need to build FST to represent each expansion step, *e.g.*,

    ```
    1     2     THE
    2     3     DOG
    3
    ```

  - With composition operation, we're done!
- Composition is efficient.
- Context-dependent expansion can be handled effortlessly.

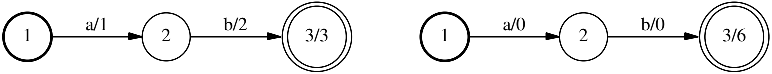
What About Those Probability Thingies?

- *e.g.*, to hold language model probs, transition probs, etc.
- FSM's ⇒ *weighted* FSM's
  - WFSA's, WFST's
- Each arc has a score or *cost*.
  - So do final states.

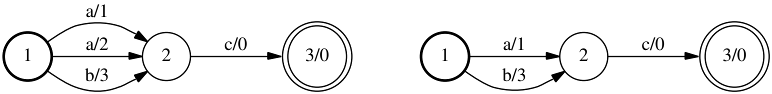


Arc Costs *vs.* Probabilities

- Typically, we take costs to be negative log probabilities.
  - Costs can move back and forth along a path.
  - The cost of a path is sum of arc costs plus final cost.



- If two paths have same labels, can be combined into one.
  - Typically, use min operator to compute new cost.

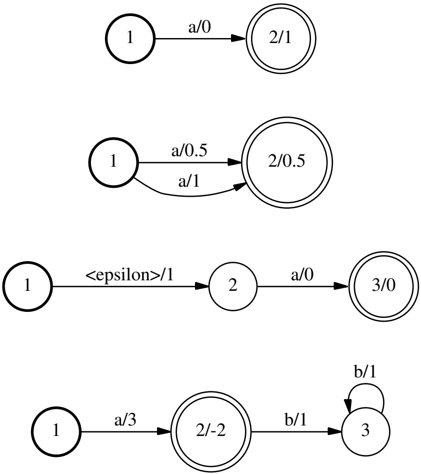


- Operations $(+, \min)$ form a *semiring* (the *tropical* semiring).
  - Other semirings are possible.

The Meaning of Life

- WFSA: a list of (unique) string and cost pairs $\{(i_1 \cdots i_N, c)\}$.
- WFST: a list of triples $\{(i_1 \cdots i_N, o_1 \cdots o_M, c')\}$.

Which Is Different From the Others?

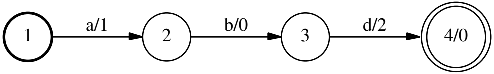


Weighted Composition

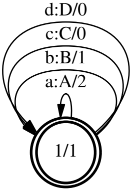
- Composing WFSA A with WFST T to get WFSA $A \circ T$.
- If $(i_1 \cdots i_N, c) \in A$ and ...
- $(i_1 \cdots i_N, o_1 \cdots o_M, c') \in T$, ...
- Then, $(o_1 \cdots o_M, c + c') \in A \circ T$.
- Combine costs for all different ways to produce same $o_1 \cdots o_M$.

Weighted Composition

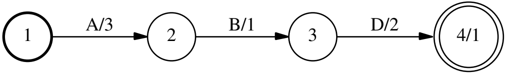
A



T



$A \circ T$



Weighted Graph Expansion

- Start with weighted FSA representing language model.
- Use composition to apply weighted FST for each level of expansion.
  - Scores/logprobs will be accumulated.
  - Log probs may move around along paths.
  - All that matters for Viterbi is total score of paths.

Recap: Composition

- Like `sed`, but can operate on all paths in a lattice simultaneously.
- Rewrite symbols as other symbols.
  - *e.g.*, rewrite words as phone sequences (or vice versa).
- Context-dependent rewriting of symbols.
  - *e.g.*, rewrite CI phones as their CD variants.
- Add in new scores.
  - *e.g.*, language model lattice rescoring.
- Restrict the set of allowed paths/intersection.
  - *e.g.*, find all paths in lattice containing word NOODGE.
- Or all of the above at once.

Road Map

- Part I: The LVCSR acoustic model.
- Part II: Acoustic model training for LVCSR.
- Part III: Decoding for LVCSR (inefficient).
  - Part IV: Introduction to finite-state transducers.
- Part V: Search (Lecture 8).
  - Making decoding for LVCSR efficient.

Course Feedback

1. Was this lecture mostly clear or unclear? What was the muddiest topic?
2. Other feedback (pace, content, atmosphere)?