ELEN E6884/COMS 86884 Speech Recognition Lecture 7

Michael Picheny, Ellen Eide, Stanley F. Chen
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
{picheny,eeide,stanchen}@us.ibm.com

20 October 2005

Administrivia

- main feedback from last lecture
  - everything was pretty clear (eventually)
- Ellen will be away for a while
  - preparing for season 4 tryouts of *Nashville Star*
- sample answers to Lab 1 posted
  - in same directory you got Lab 1 files from
- Lab 2 due Sunday midnight
- Lab 3 out Monday?

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

The Fundamental Equation of Speech Recognition

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

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

Outline

- Unit I: you do not talk about Unit I
- Unit II: acoustic model training for LVCSR
- Unit III: decoding for LVCSR (inefficient)
  - Unit IV: introduction to finite-state transducers
- Unit V: search (lecture 8)
  - making decoding for LVCSR efficient

Unit II: Acoustic Model Training for LVCSR

Small vocabulary training — Lab 2

- small model
  - 102 HMM states spread over 11 word models
  - 102 12-dimensional Gaussians
  - $\Rightarrow 102 \times 12 \times 2 = 2448$ parameters
- simple training recipe
  - flat start: mean 0, variance 1
  - run a bunch of iterations of Forward-Backward training
  - done!

Acoustic Model Training

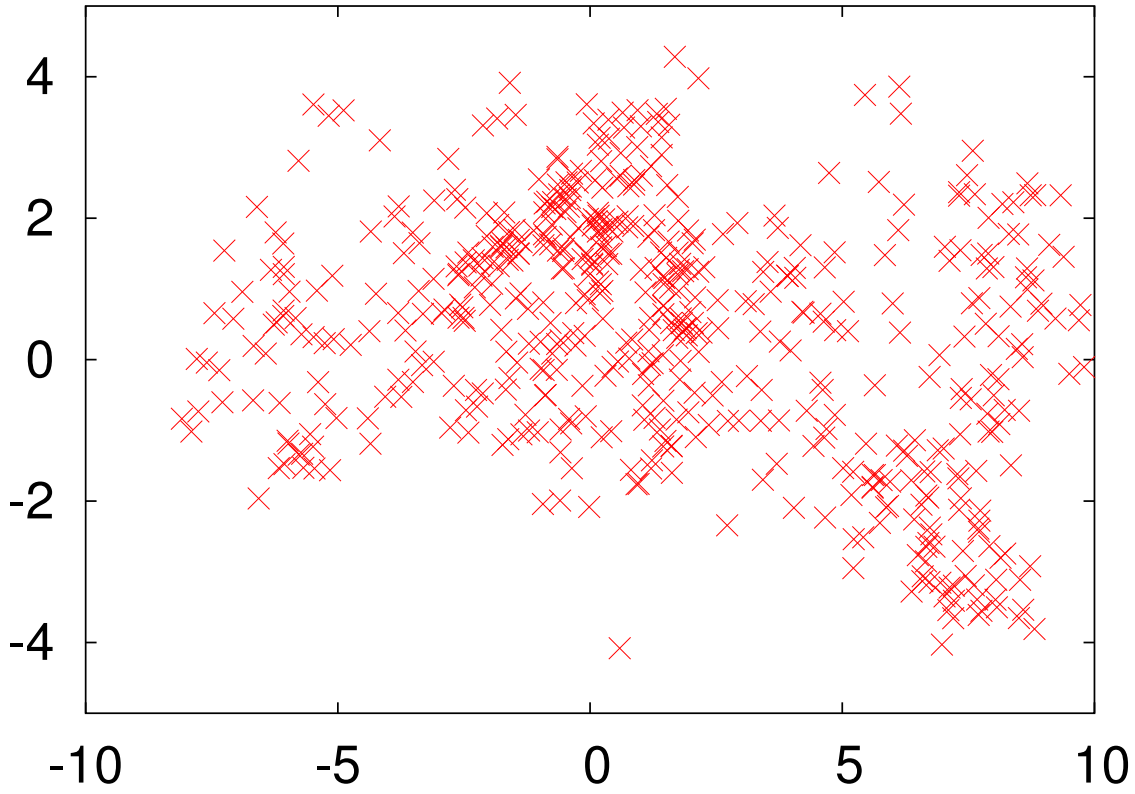
What happens when we train more complex acoustic models?

- single Gaussians $\Rightarrow$ Gaussian mixture models (GMM's)
- isolated speech $\Rightarrow$ continuous speech
- word models $\Rightarrow$ context-dependent (CD) phone models
- 2500 Gaussian parameters $\Rightarrow$ tens of millions of Gaussian parameters
- flat start and FB?

Case Study: Training a Mixture of Two 2-D Gaussians

The Data: real live acoustic features

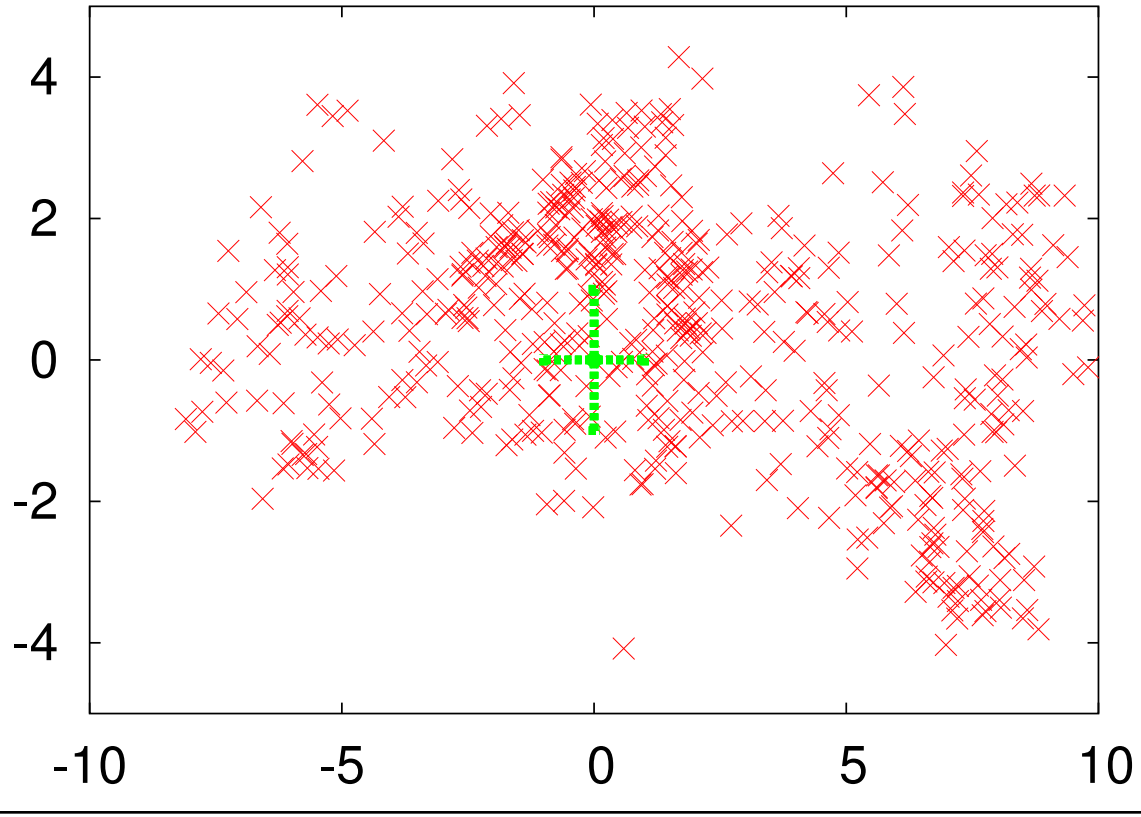
- front end from lab 1; take first two dimensions; 546 frames



Training a Mixture of Two 2-D Gaussians

Flat start?

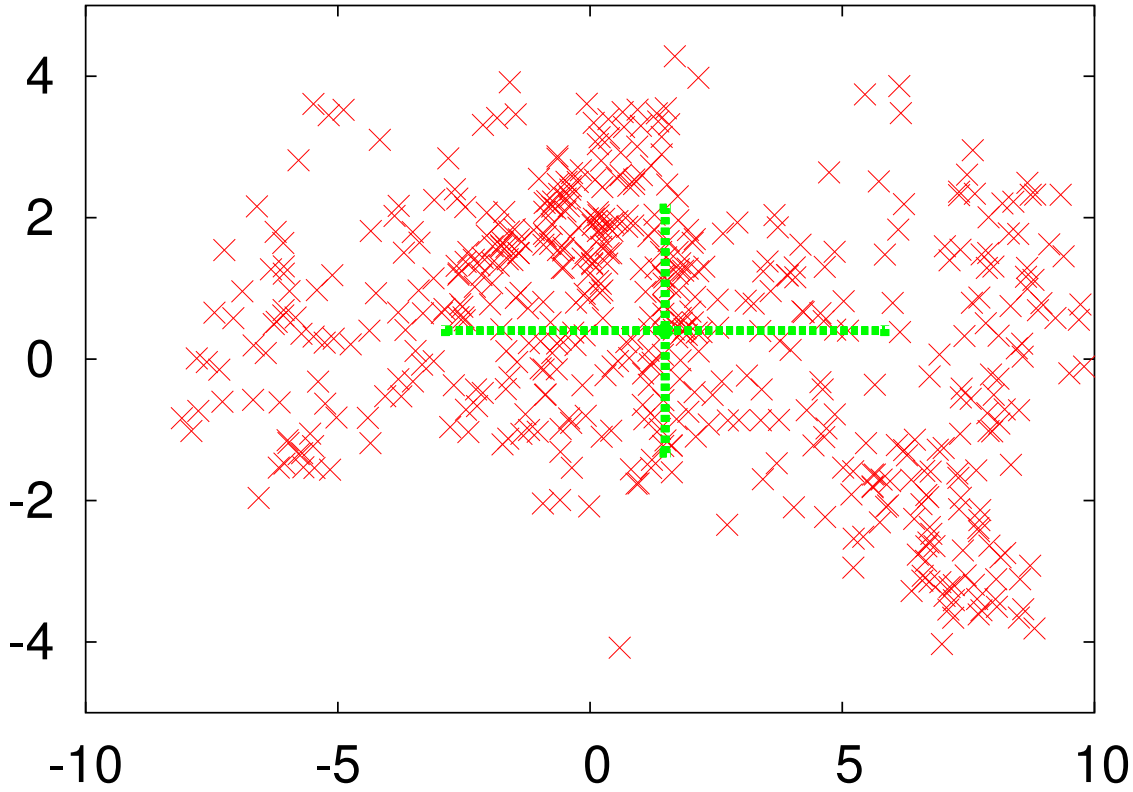
- initialize mean of each Gaussian to 0, variance to 1
- what do you think will happen?



Training a Mixture of Two 2-D Gaussians

"*At the Mr. O level, symmetry is everything.*"

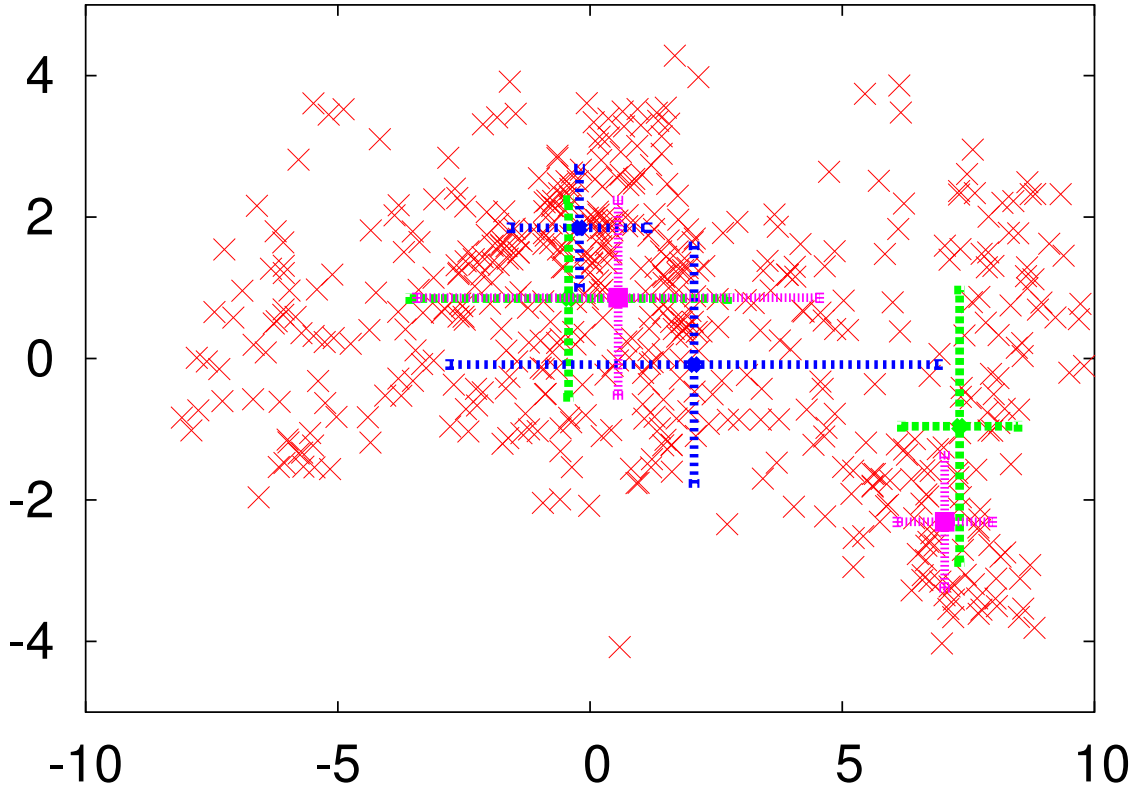
- at the GMM level, symmetry is a bad idea.



Training a Mixture of Two 2-D Gaussians

Random seeding?

- picked 8 random starting points $\Rightarrow$ 3 different optimum found
- training is not simple even for simple models



Training Hidden Models

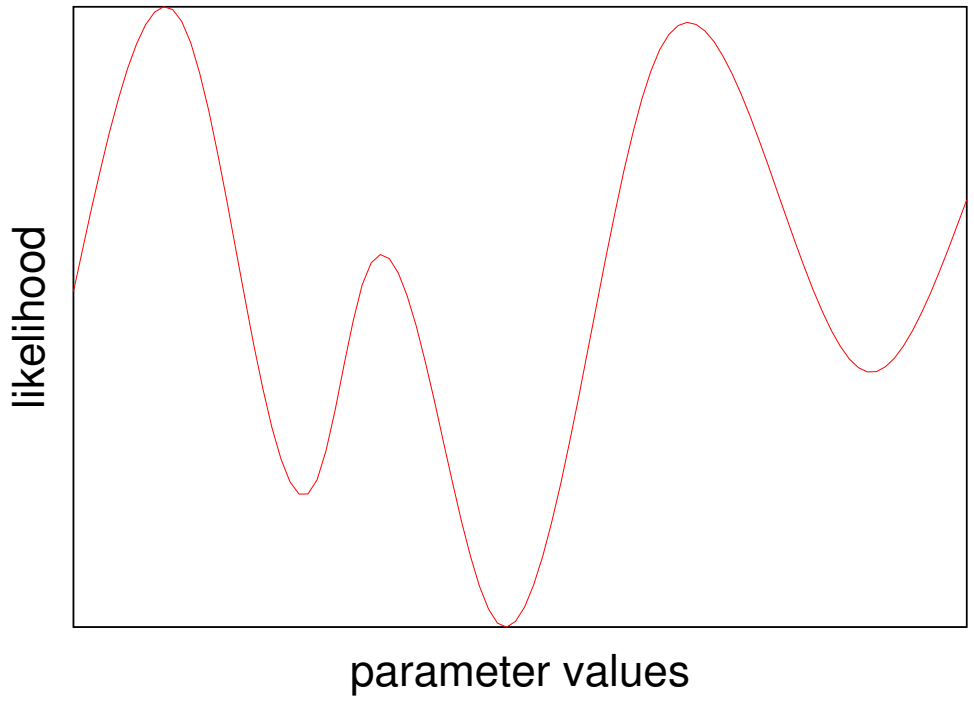
(MLE) training of models with hidden variables has local minima

- example: GMM
  - hidden quantity: for each feature vector in training data, which Gaussian in mixture generated it
- example: HMM
  - hidden quantity: alignment; for each frame in training data, which arc generated it

Gradient Descent and Local Minima

FB training does hill-climbing/gradient descent

- finds “nearest” optimum to where you started
- picking a good starting point is key
- chicken and egg problem



Secrets of Acoustic Model Training, Uncovered!

Unit overview

- discuss training process in more depth
- reveal strategies for finding ML parameter estimates for complex models
  - discovered via sweat and tears
  - not true ML estimates, but as close as we can get
  - art, not science
- in practice, training is tortuous multistage process
  - use simpler models to bootstrap more complex models

Aside: Not Truly Maximum Likelihood

- variance flooring
  - don't let variances go to 0 $\Rightarrow$ infinite likelihood
- just as LM's need to be smoothed or *regularized*
  - so do acoustic models
  - penalize undesirable parameter values/unsmooth models
  - variance flooring is poor person's regularization

Baby Steps

Let's start simple and consider more complex models in turn

- from word models; single Gaussians; isolated words . . .
- to context-dependent phone models; GMM's; continuous words

Single Gaussian Word Models, Isolated Word

- 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!

Single Gaussian Word Models, Isolated Word

Why does this work?

- we believe there's a huge local minima 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)
- another perspective
  - the model doesn't have enough freedom to screw up
  - the only way it can achieve a good likelihood is . . .
    - if the Gaussians for a particular phone (*e.g.*, AH) . . .
    - actually model the acoustic realizations of that phone

Bootstrapping Big Models From Small Models

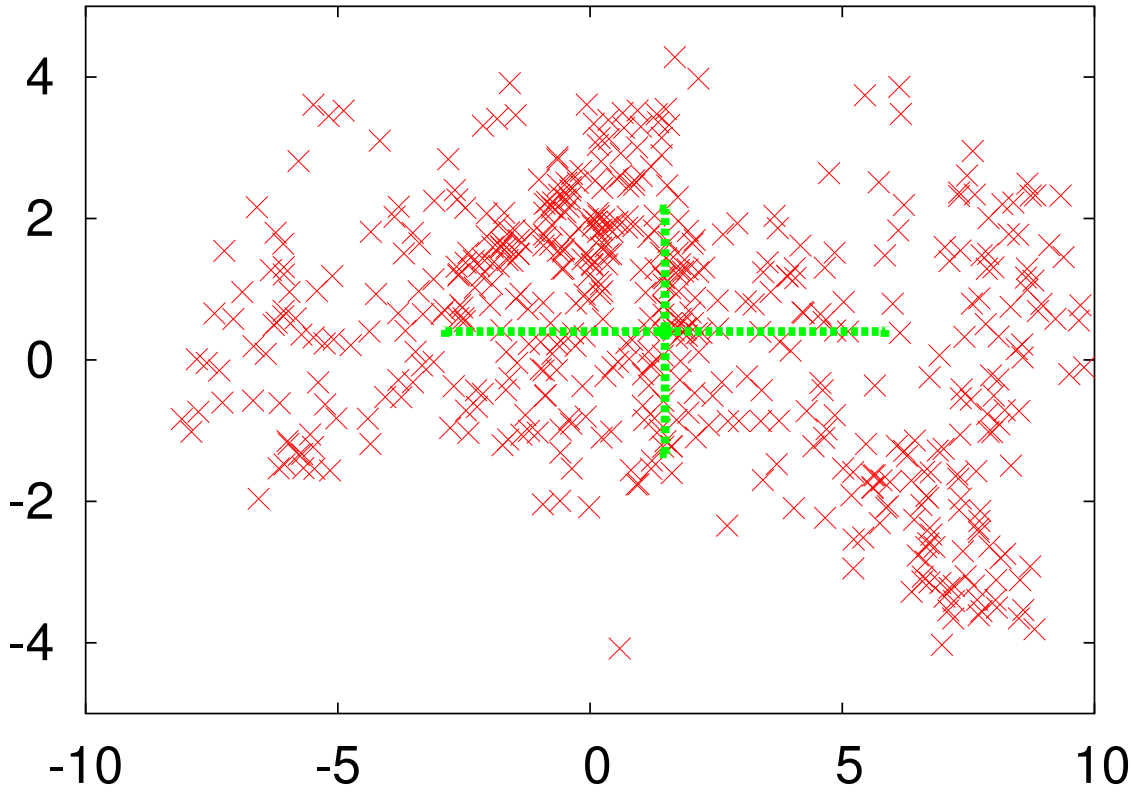
- how can we train more complex models . . .
  - where flat/random start will almost certainly do poorly?
- start with a model simple enough that a flat start works
- then, can we use this simple model . . .
  - to give us hints on how to seed the parameters of a larger model?
- if so, can iteratively build more and more complex models
- case study: training mixtures of Gaussians
  - recursive mixture splitting
  - k-means clustering

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: n-component Gaussian mixture gives good hints on how to seed $2n$-component Gaussian mixture

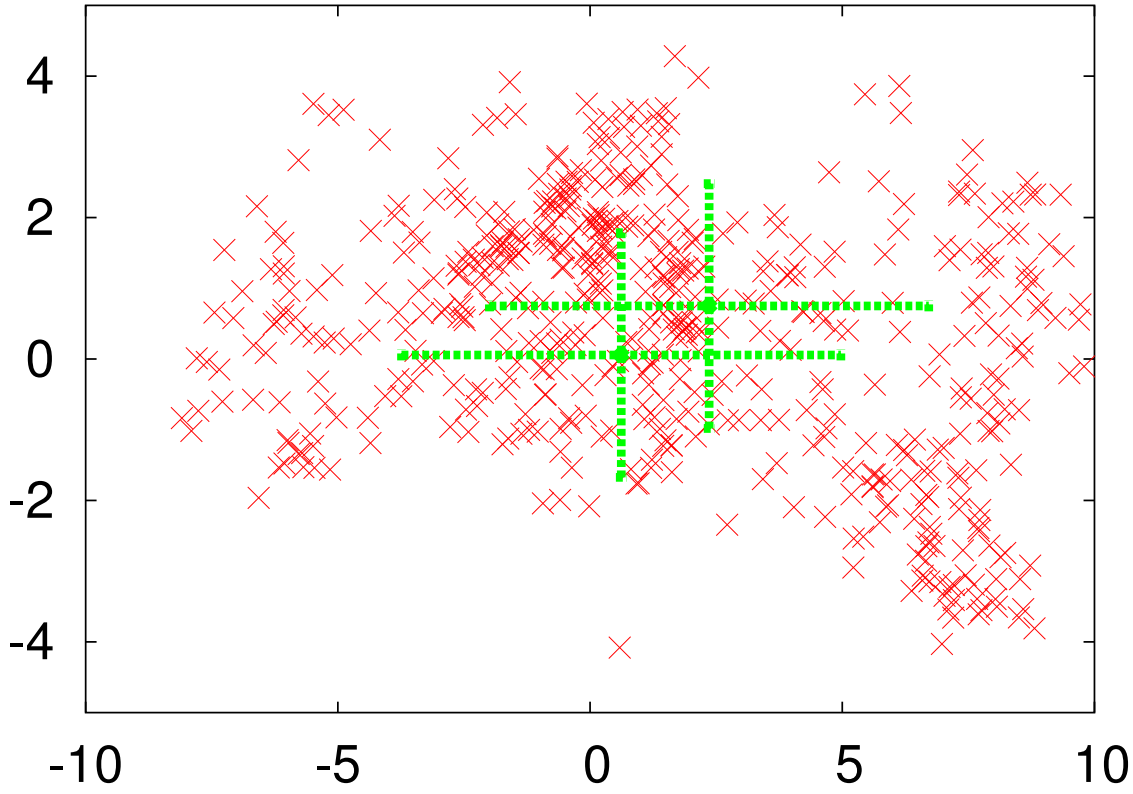
Mixture Splitting Example

- train single Gaussian



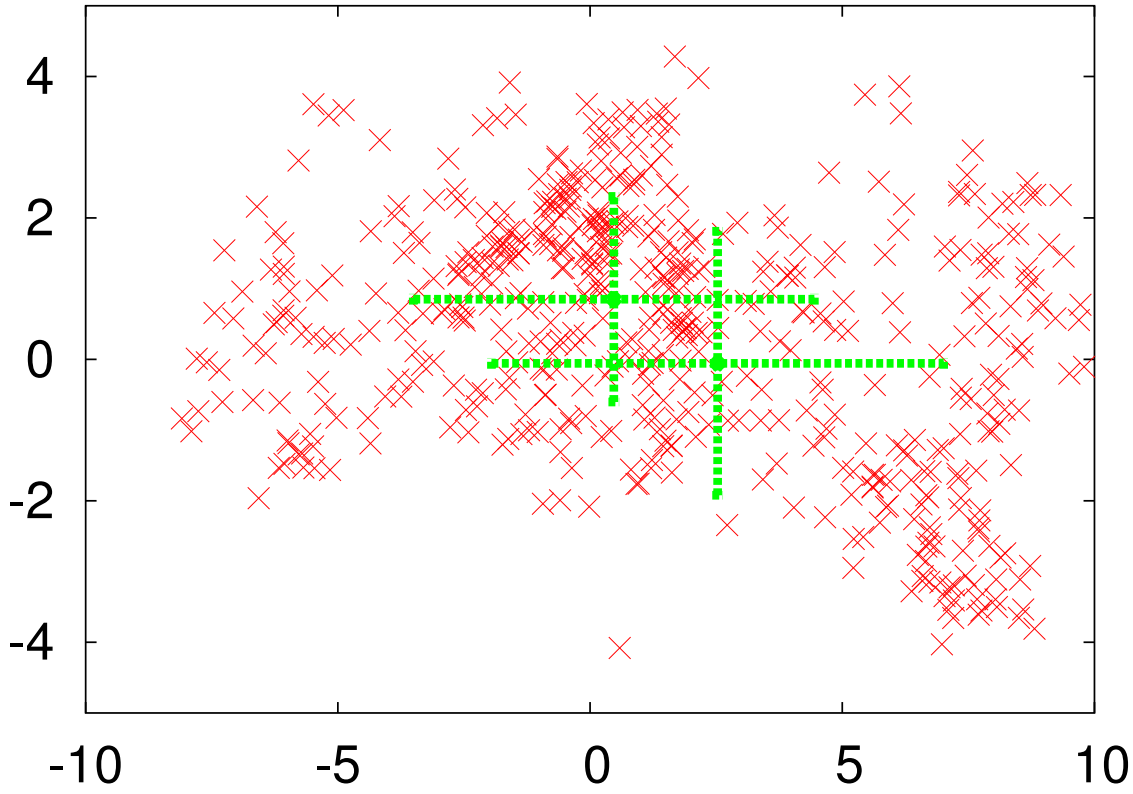
Mixture Splitting Example

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



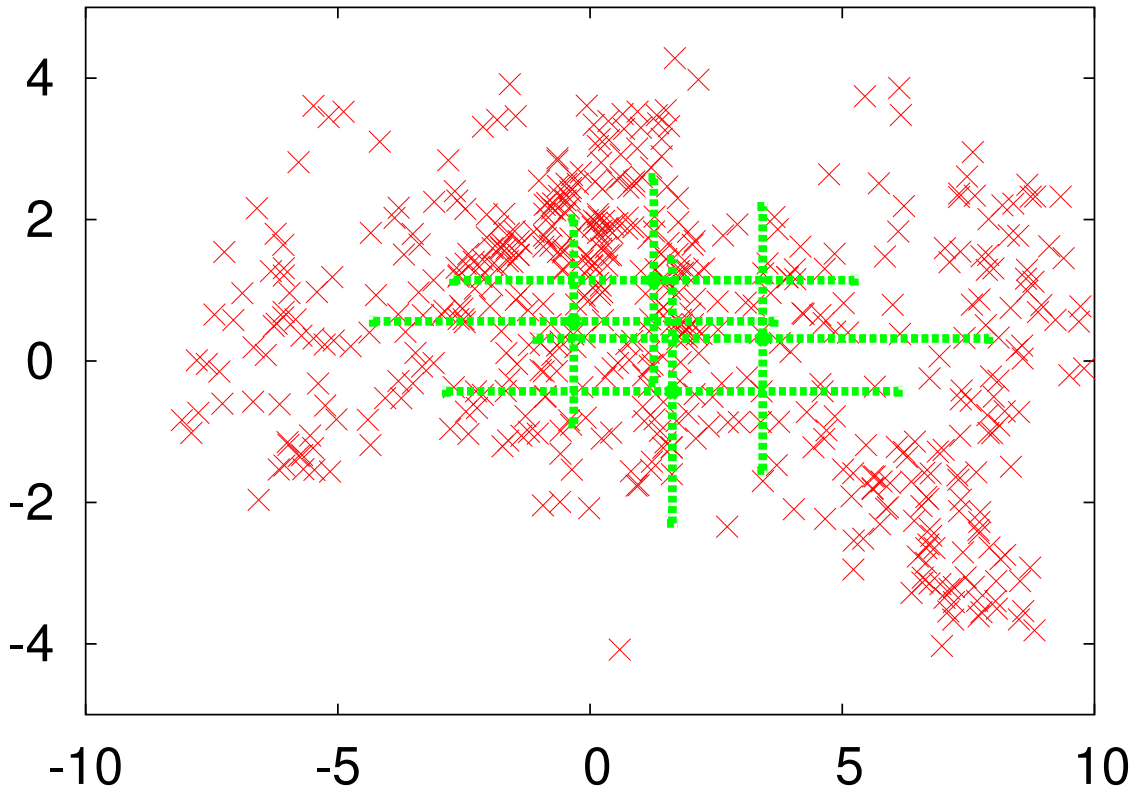
Mixture Splitting Example

- train, yep



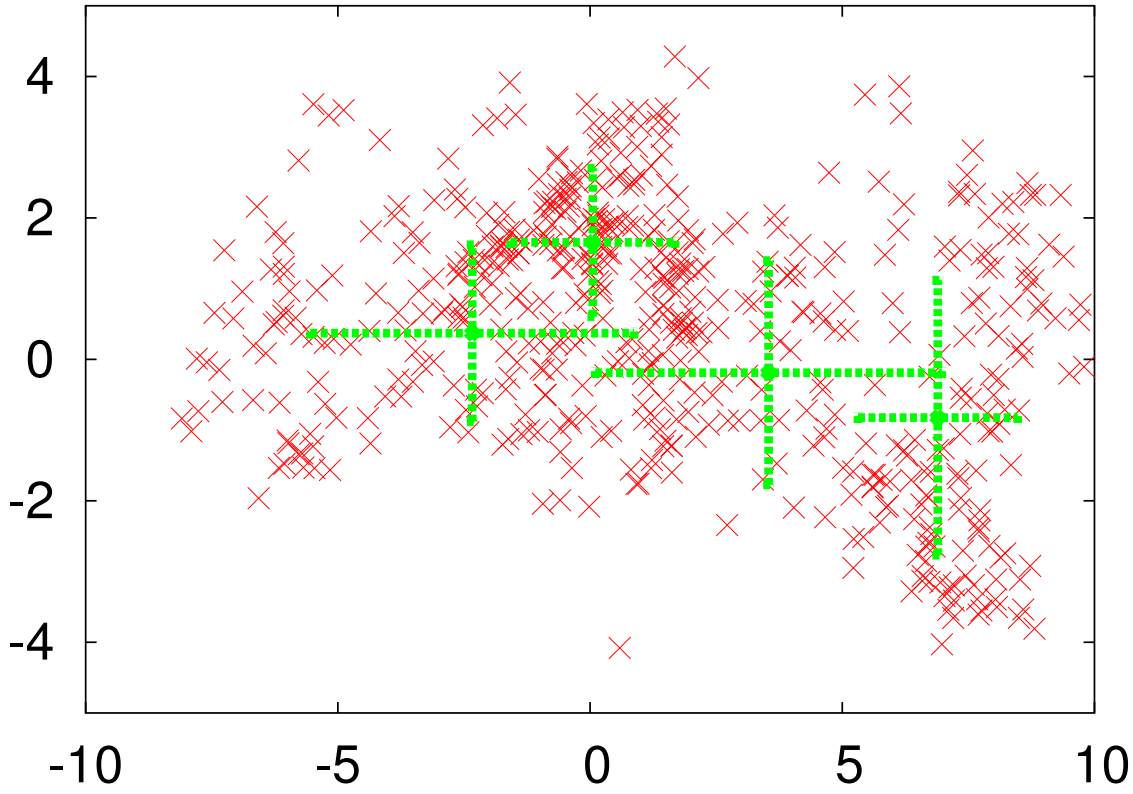
Mixture Splitting Example

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



Mixture Splitting Example

- train, yep



Using Mixture Splitting in Acoustic Model Training

- train model where each output distribution is single Gaussian (à la Lab 2)
- split Gaussians in each output distribution simultaneously
- train whole model with FB
- repeat

Another Seeding Method: Use Automatic Clustering

- instead of recursive divide-and-conquer method . . .
- use clustering algorithm on data to find desired number of cluster centers all at once
  - use cluster centers to seed Gaussian means
  - initialize variances to constant
- (discard Gaussians with insufficient counts)

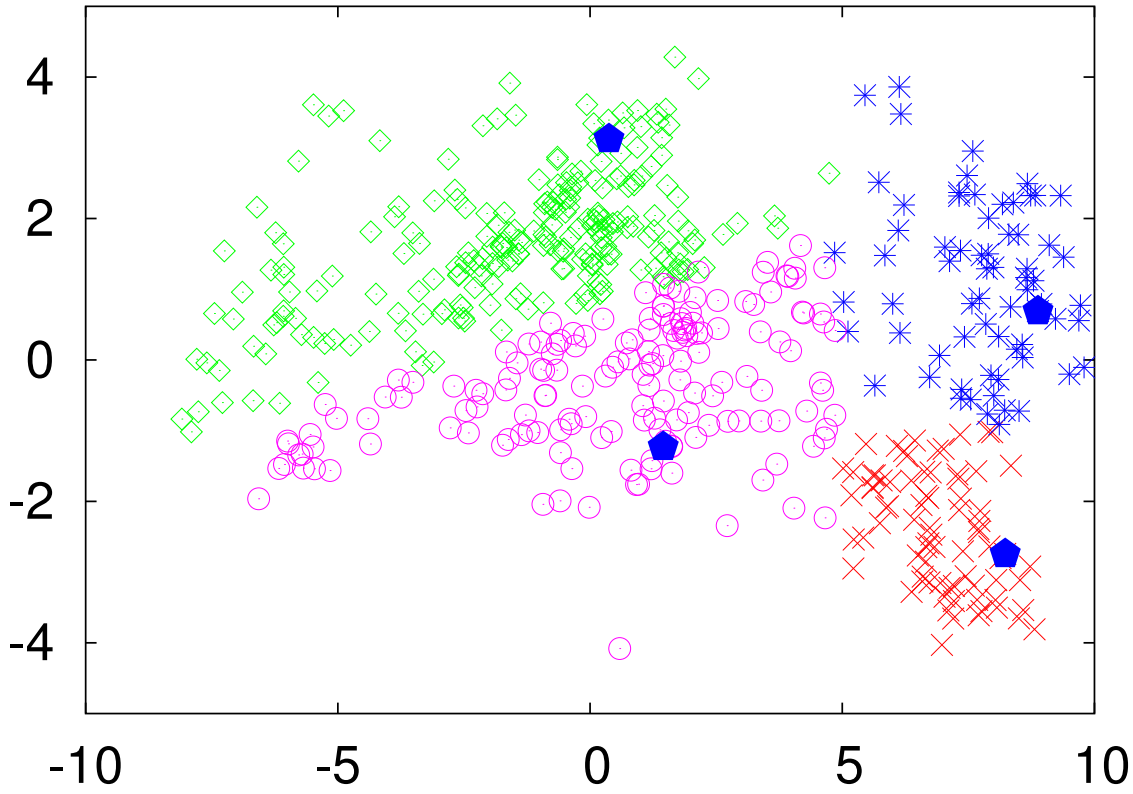
k-Means Clustering

Simple and effective clustering algorithm

- 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

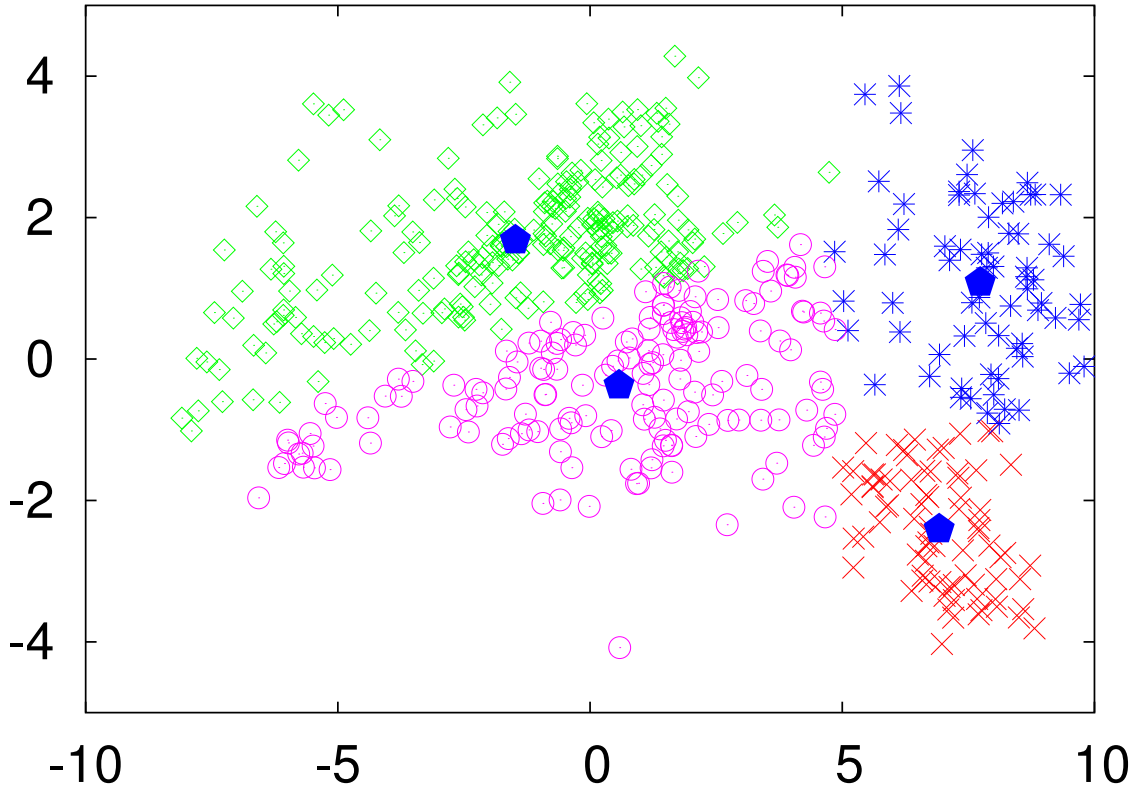
k-Means Example

- pick random cluster centers; assign each point to nearest center



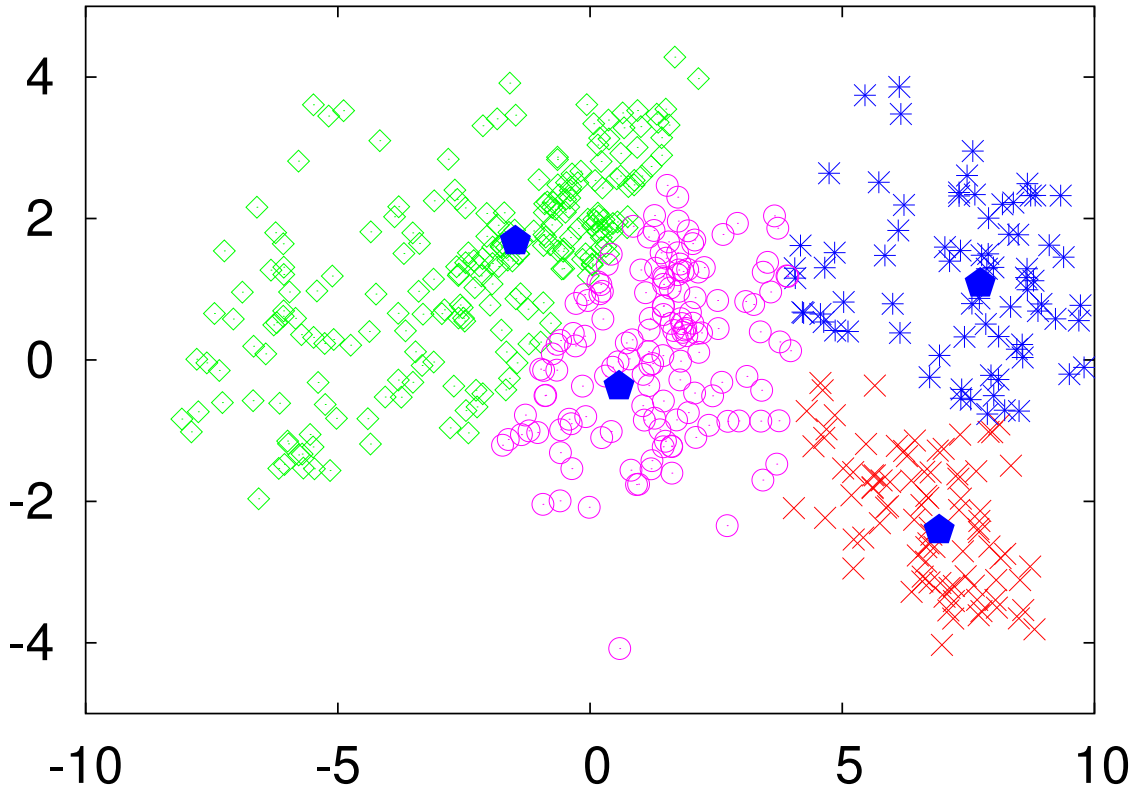
k-Means Example

- recompute cluster centers



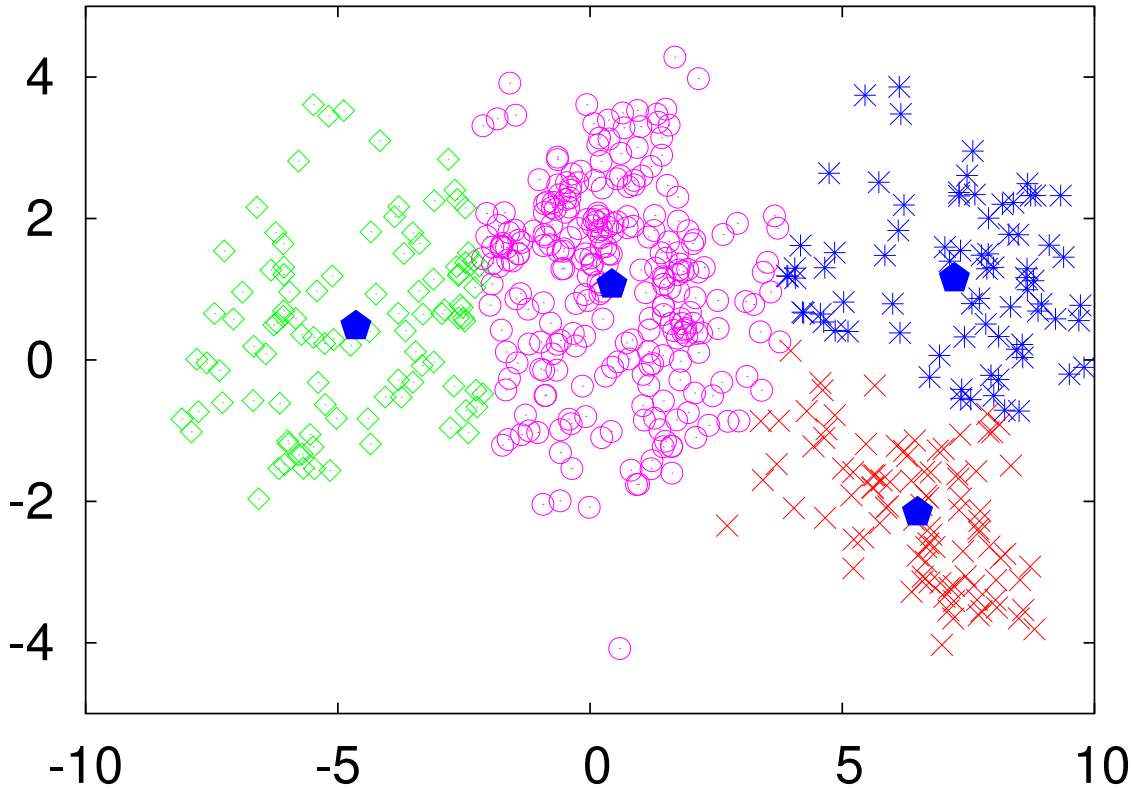
k-Means Example

- assign each point to nearest center



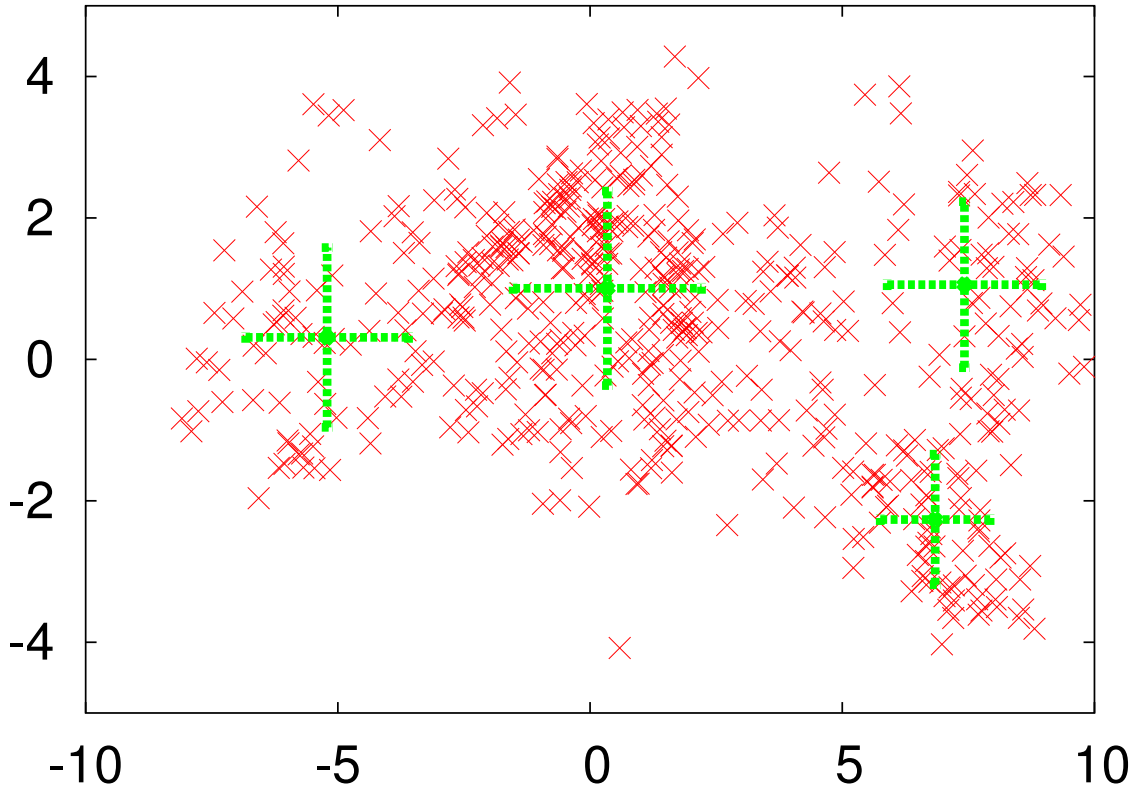
k-Means Example

- repeat until convergence

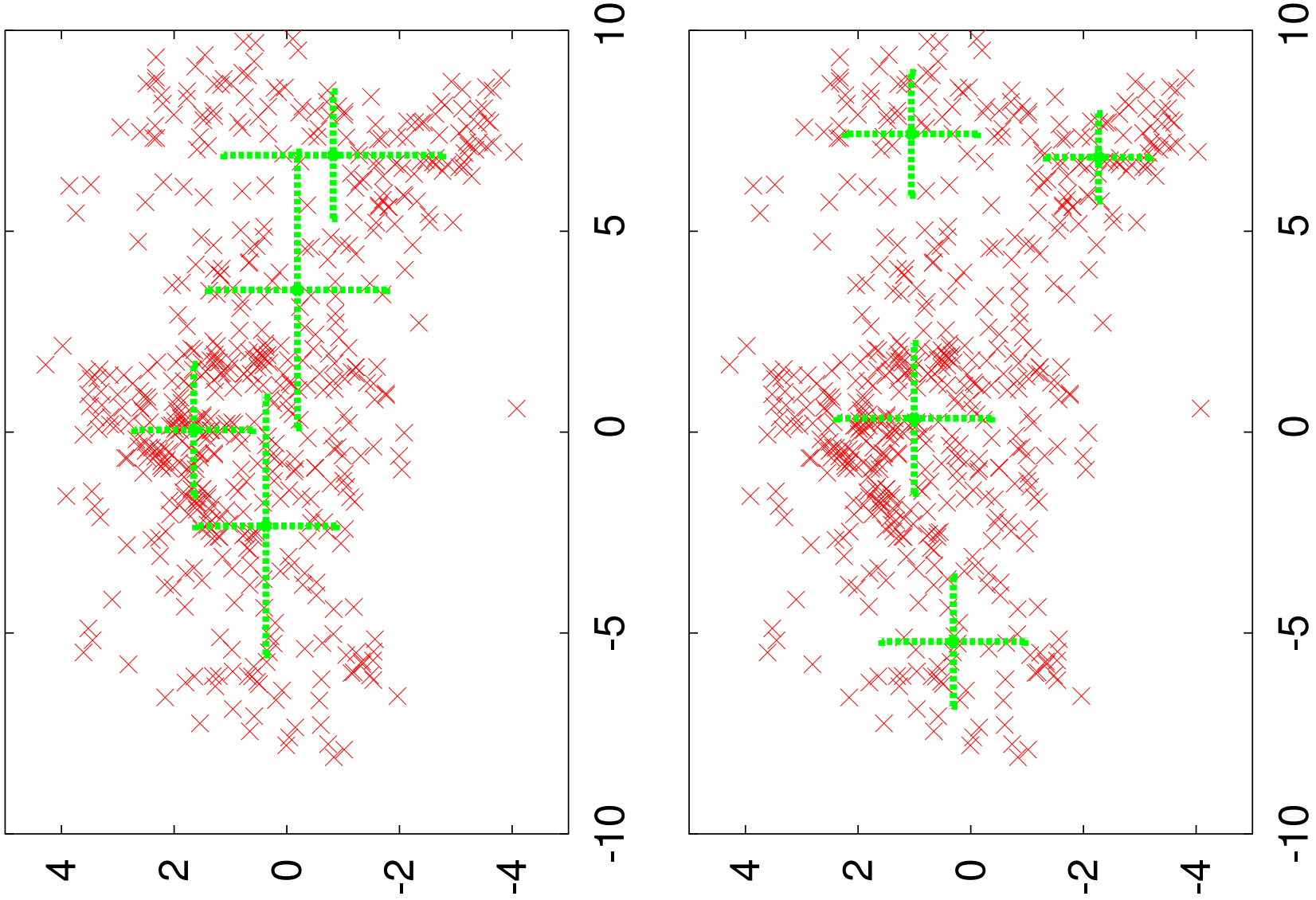


k-Means Example

- use centers as means of Gaussians; train, yep



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



Technical Aside: k-Means Clustering

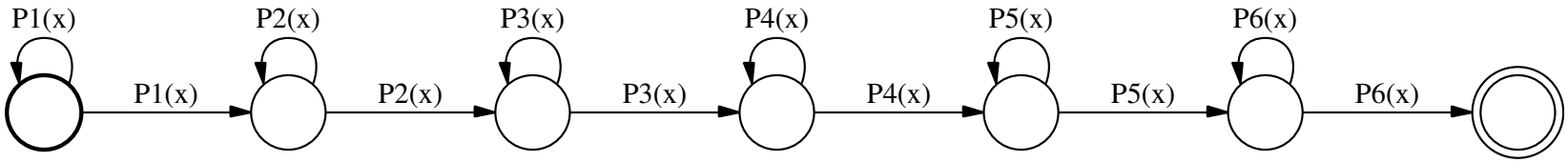
- when using Euclidean distance to compute "nearest" center . . .
- k-means clustering is equivalent to . . .
  - seeding k-component GMM means with the k initial centers
  - doing "hard" GMM update
    - instead of assigning true posterior to each Gaussian in update . . .
    - assign "posterior" of 1 to most likely Gaussian and 0 to the others
  - keeping variances constant

Using k-Means Clustering in Acoustic Model Training

- for each GMM/output distribution, use k-means clustering . . .
  - on acoustic feature vectors “associated” with that GMM . . .
  - to seed means of that GMM
- huh?
  - how to decide which frames belong to which GMM?
  - we are told which word (HMM) belongs to each training utterance
    - but we aren’t told which HMM arc (output distribution) belongs to each frame
- how can we compute this?

Forced Alignment

- Viterbi algorithm
  - given acoustic model, finds most likely alignment of HMM to data
  - not perfect, but what can you do?



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 . . .
  - for seeding means for GMM's in new model
  - use best existing model you have available!
    - alignment will only be as good as model

Lessons: Training GMM's

- hidden models have local minima galore!
- smaller models can help seed larger models
  - mixture splitting
    - use n-component GMM to seed $2n$-component GMM
  - k-means
    - use existing model to provide GMM$\Leftrightarrow$frame alignment
- heuristics have been developed that work OK
  - mixture splitting and k-means are comparable
    - but no one believes these find global optima, even for relatively small problems
  - these are not the last word!

Single Gaussians $\Rightarrow$ GMM's

The training recipe so far

- train single Gaussian models (flat start; many iterations of FB)
- do mixture splitting, say
  - split each Gaussian in two; many iterations of FB
  - repeat until desired number of Gaussians per mixture

Unit II: Acoustic Model Training for LVCSR

What's next?

- single Gaussians $\Rightarrow$ Gaussian mixture models (GMM's)
- isolated speech $\Rightarrow$ continuous speech
- word models $\Rightarrow$ context-independent (CI) phone models
- CI phone models $\Rightarrow$ context-dependent (CD) phone models

From Isolated to Continuous Speech

- isolated speech with word models
  - train each word HMM using only instances of that word
- continuous speech
  - don’t have instances of individual words nicely separated out
  - don’t know when each word begins and ends in an utterance
- what to do?

From Isolated to Continuous Speech

Strategy A (Viterbi-style training)

- do forced alignment
  - for each training utterance, build HMM by . . .
    - concatenating word HMM's for words in reference transcript
  - do Viterbi algorithm; recover best alignment
  - see board
- snip each utterance into individual words
  - reduces to isolated word training
- what are possible issues with this approach?

From Isolated to Continuous Speech

Strategy B

- instead of snipping the concatenated word HMM and snipping the acoustic feature vectors . . .
  - and running FB on each word HMM+segment separately . . .
  - what if we just run FB on the whole darn thing!?
- does this make sense?
  - like having an HMM for each word sequence rather than for each word . . .
  - where parameters for all instances of same word are *tied*
  - analogy: like using phonetic models for isolated speech
    - each word (phone sequence) has its own HMM . . .
    - where parameters for all instances of same phone are tied

Pop Quiz

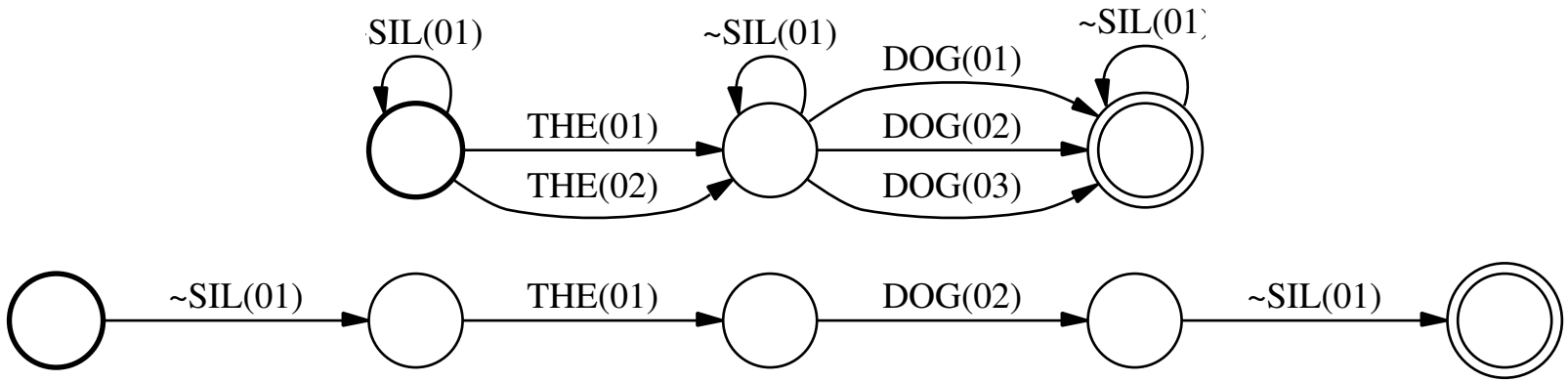
- To do one iteration of FB, which strategy is faster?
  - Hint: what is the time complexity of FB?
- Which strategy is less prone to local minima?
- in practice, both styles of strategies are used
  - including an extreme version of Strategy A

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`?
- how can we correctly construct the HMM for an utterance?
  - where do we insert the silence HMM?
  - which pronunciation variant to use for each word?
    - if have different HMM's for different pronunciations of a word

Pronunciation Variants, Silence, and Stuff

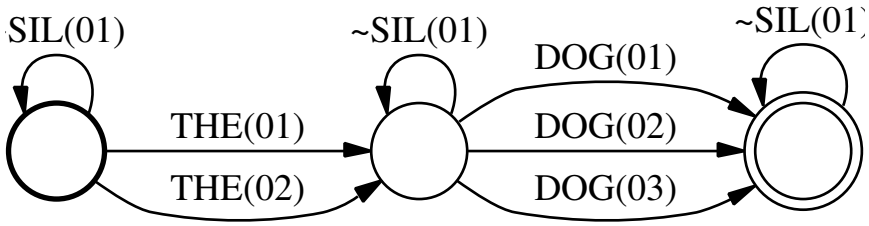
- that is, the human-produced transcript is incomplete
  - how can we produce a more complete transcript?
- Viterbi decoding!
  - build HMM accepting all word (HMM) sequences consistent with reference transcript
  - compute best path/word HMM sequence



Pronunciation Variants, Silence, and Stuff

Where does the initial acoustic model come from?

- train initial model without silence; single pronunciation per word
- use HMM containing all alternatives directly in training (*e.g.*, Lab 2)
  - not clear what interpretation is, but works for bootstrapping



Isolated Speech $\Rightarrow$ Continuous Speech

The training recipe so far

- train an initial GMM system (Lab 2 stopped here)
  - same recipe as before, except create HMM for each training utterance by concatenating word HMM's
- use initial system to refine reference transcripts
  - select pronunciation variants, where silence occurs
- do more FB on initial system or retrain from scratch
  - using refined transcripts to build HMM's

Unit II: Acoustic Model Training for LVCSR

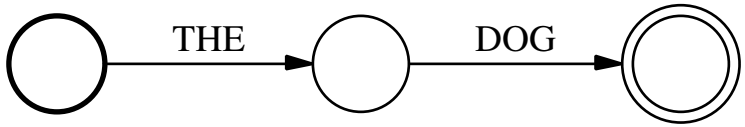
What's next?

- single Gaussians $\Rightarrow$ Gaussian mixture models (GMM's)
- isolated speech $\Rightarrow$ continuous speech
- word models $\Rightarrow$ context-independent (CI) phone models
- CI phone models $\Rightarrow$ context-dependent (CD) phone models

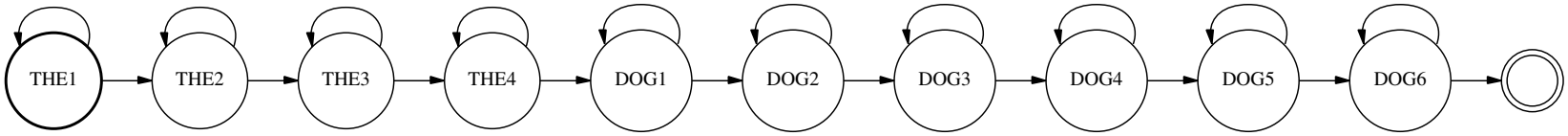
Word Models

HMM/graph expansion

- reference transcript



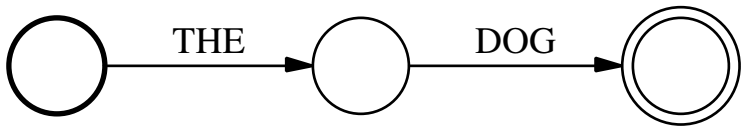
- replace each word with its HMM



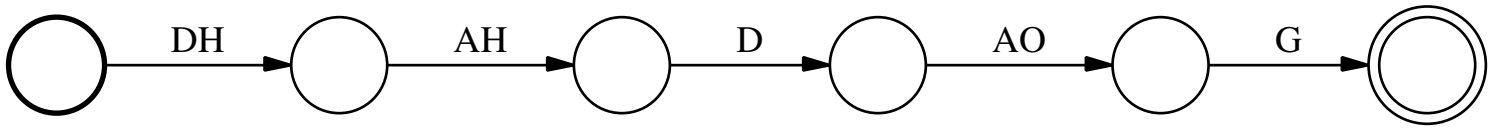
Context-Independent Phone Models

HMM/graph expansion

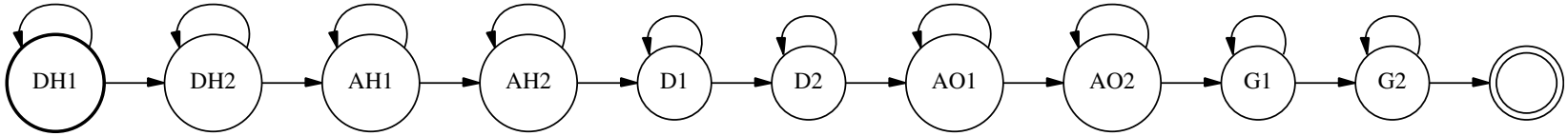
- reference transcript



- pronunciation dictionary
  - maps each word to a sequence of phonemes



- replace each phone with its HMM



Word Models $\Rightarrow$ Context-Independent Phone Models

Changes

- 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?
- how the HMM for each training utterance is created

Word Models $\Rightarrow$ Context-Independent Phone Models

The training recipe so far

- build pronunciation dictionary for all words in training set
- train an initial GMM system
- use initial system to refine reference transcripts
- do more FB on initial system or retrain from scratch

Unit II: Acoustic Model Training for LVCSR

What's next?

- single Gaussians $\Rightarrow$ Gaussian mixture models (GMM's)
- isolated speech $\Rightarrow$ continuous speech
- word models $\Rightarrow$ context-independent (CI) phone models
- CI phone models $\Rightarrow$ context-dependent (CD) phone models

CI $\Rightarrow$ CD Phone Models

- context-independent phone models
  - there are $\sim$50 phonemes
  - each has a $\sim$3 state HMM $\Rightarrow$ $\sim$150 CI HMM states
  - each CI HMM state has its own GMM $\Rightarrow$ $\sim$150 GMM's
- context-dependent models
  - each of the $\sim$150 HMM states now has a set of 1–100 GMM's attached to it
  - which of the 1–100 GMM's to use is determined by the phonetic context ...
    - by using a decision tree
    - *e.g.*, for first state of phone AX, if DH to left and stop consonant to right, then use GMM_{37}, else ...

Context-Dependent Phone Models

Notes

- not one decision tree per phoneme, but one per phoneme *state*
  - better model of reality
  - GMM for first state in HMM depends on left context mostly
  - GMM for last state in HMM depends on right context mostly
- terminology
  - *triphone* model — look at ± 1 phones of context
  - *quinphone* model — look at ± 2 phones of context
  - also, *septaphone* and 11-phone 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	1–100	1k–10k	10k–300k

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

Building a Triphone Phonetic Decision Tree

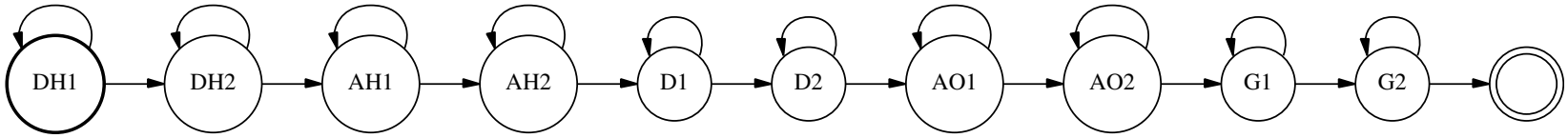
- in a CI model, consider the GMM for a state, *e.g.*, AH_1
  - this is a probability distribution $p(\vec{x}|\mathsf{AH}_1)$ . . .
    - over acoustic feature vectors $\vec{x}$
- context-dependent modeling assumes . . .
  - we can build better model of acoustic realizations of AH_1 . . .
  - if we condition on the surrounding phones, *e.g.*, for a triphone model, $p(\vec{x}|\mathsf{AH}_1, p_L, p_R)$
- what do we mean by better model?
- how do we build this better model?

Building a Triphone Phonetic Decision Tree

- what do we mean by better model?
  - maximum likelihood!?
  - the model $p(\vec{x}|\text{AH}_1, p_L, p_R)$ should assign a higher total likelihood than $p(\vec{x}|\text{AH}_1)$ to some data $\vec{x}_1, \vec{x}_2, \ldots$
- on what data?
  - all frames $\vec{x}$ in the training data ...
  - that correspond to the state/sound AH_1
- how do we find this data?

Training Data for Decision Trees

- forced alignment/Viterbi decoding!
- where do we get the model to align with from?
  - 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 a Triphone Phonetic Decision Tree

- build decision tree for AH_1 to optimize likelihood of acoustic feature vectors aligned to AH_1
  - predetermined question set
  - see lecture 6 slides, readings for gory details
- the CD probability distribution: $p(\vec{x}|\mathsf{leaf}(\mathsf{AH}_1, p_L, p_R))$
  - there is a GMM at each leaf of the tree
  - context-independent $\Leftrightarrow$ tree with single leaf

Goldilocks and The Three Parameterizations

Perspective

- one GMM per phone state
  - too few parameters; doesn't model the many allophones of a phoneme
- one GMM per phone state and triphone context ($\sim 50 \times 50$)
  - too many parameters; sparse data issues
- cluster triphone contexts using decision tree
  - each leaf represents a cluster of triphone contexts . . .
    - with (hopefully) similar acoustic realizations that can be modeled with single GMM
  - just right!

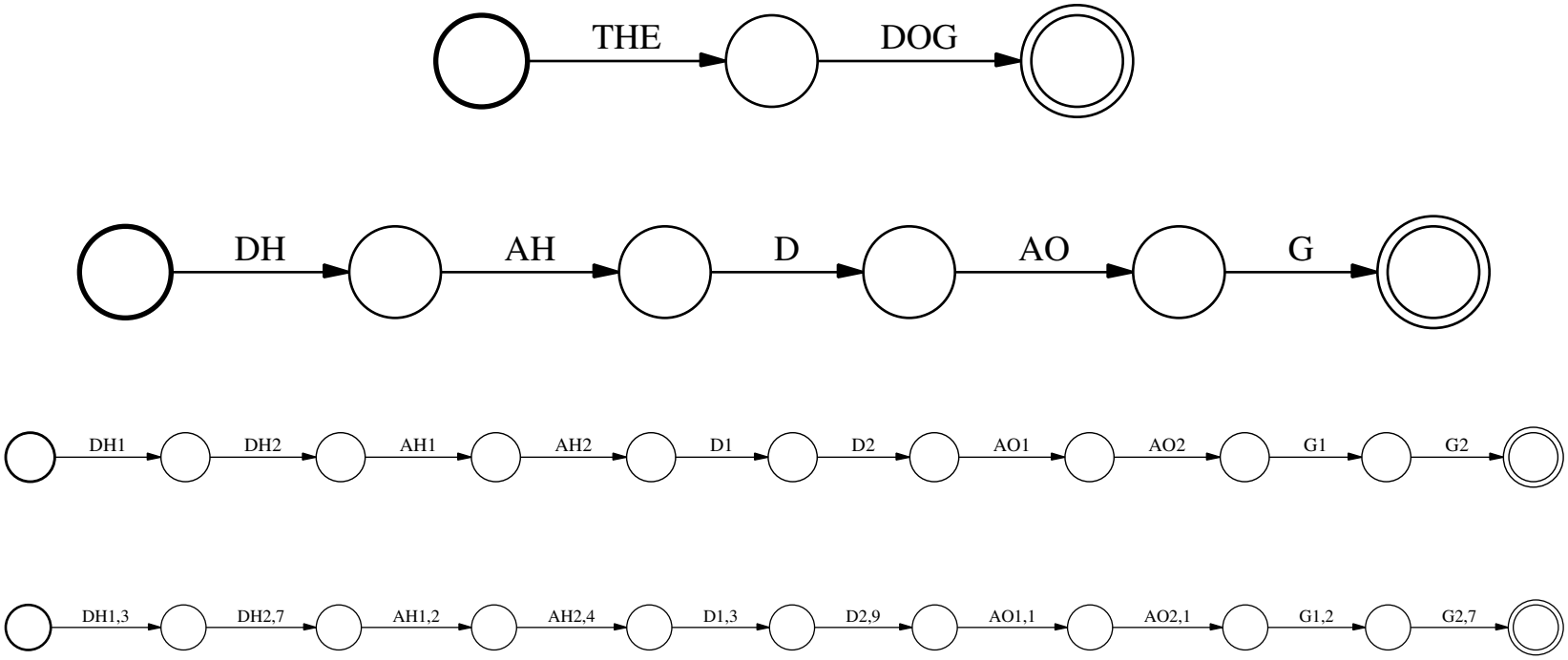
Training Context-Dependent Models

OK, let's say we have decision trees; how to train our new GMM's?

- how can we seed the context-dependent GMM parameters?
  - *e.g.*, what if we have a CI model?
  - what if we have an existing CD model but with a different tree?
- once you have a good model for a domain
  - can use to quickly bootstrap other models
  - why might this be a bad idea?

Training Context-Dependent Models

HMM/graph expansion



CI $\Rightarrow$ CD Phone Models

The training recipe so far

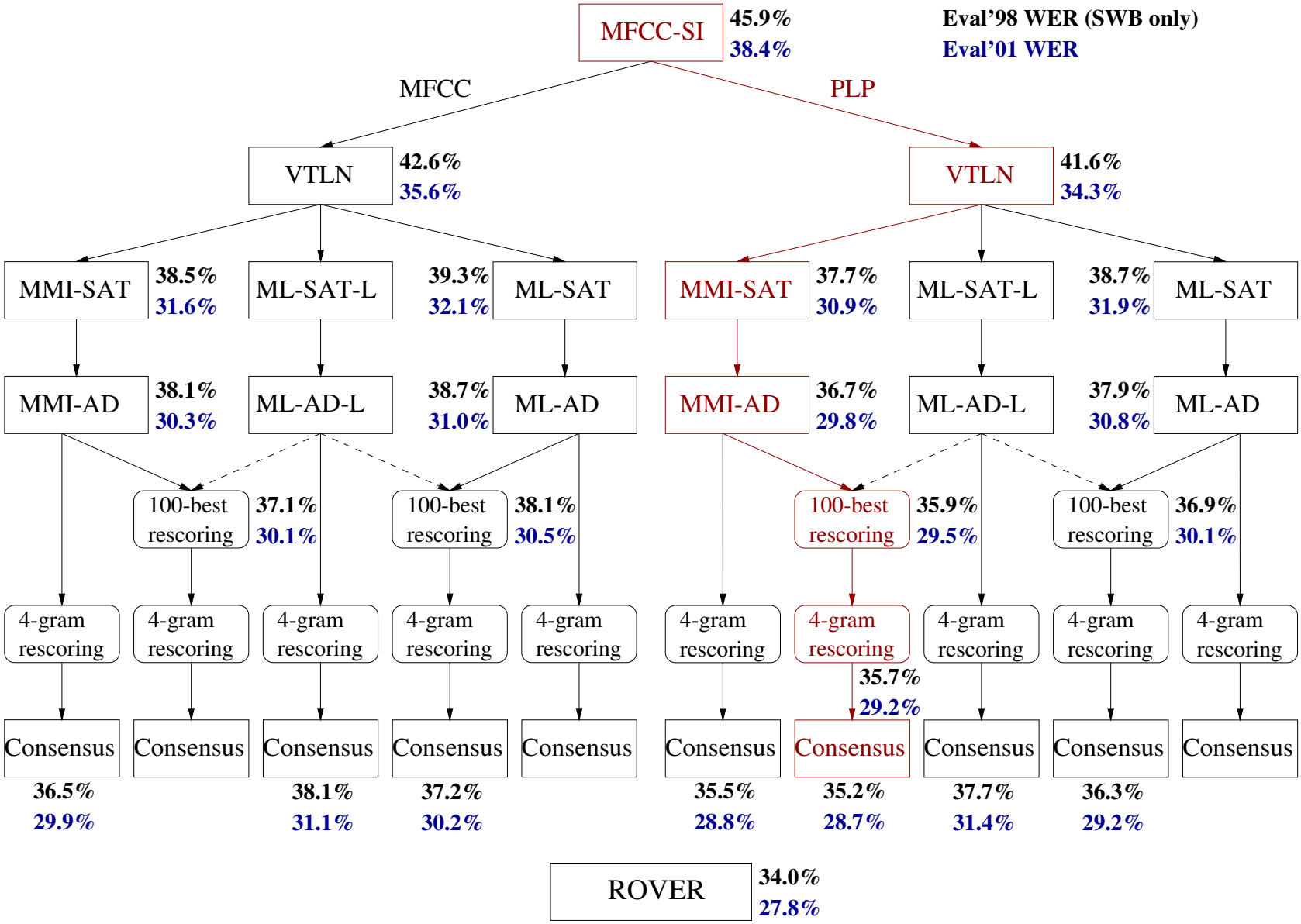
- build CI model using previous recipe
- use CI model to align training data
  - use alignment to build phonetic decision tree
- use CI model to seed CD model
- train CD model using FB

Whew, That Was Pretty Complicated!

Or not

- 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



Unit II: 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 recipes/heuristics to try to improve the minimum you end up in
    - no one best recipe
  - training is insanely complicated for state-of-the-art research models
- the good news is . . .
  - I just saved a bunch on money on my car insurance by switching to GEICO

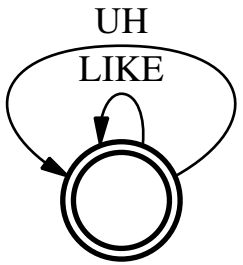
Unit III: Decoding for LVCSR (Inefficient)

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

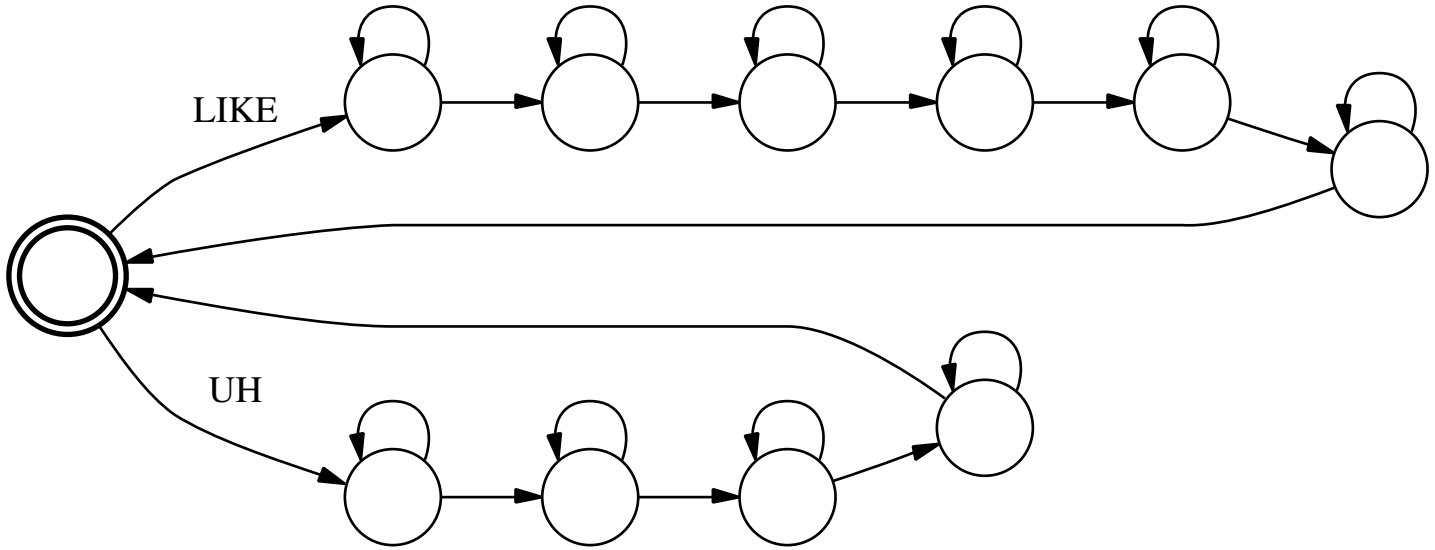
- now that we know how to build models for LVCSR . . .
  - CD acoustic models via complex recipes
  - n-gram models via counting and smoothing
- how can we use them for decoding?
  - let's ignore memory and speed constraints for now

Decoding

What did we do for small vocabulary tasks?



- take graph/FSA represent language model
  - *i.e.*, all allowed word sequences
- expand to underlying HMM

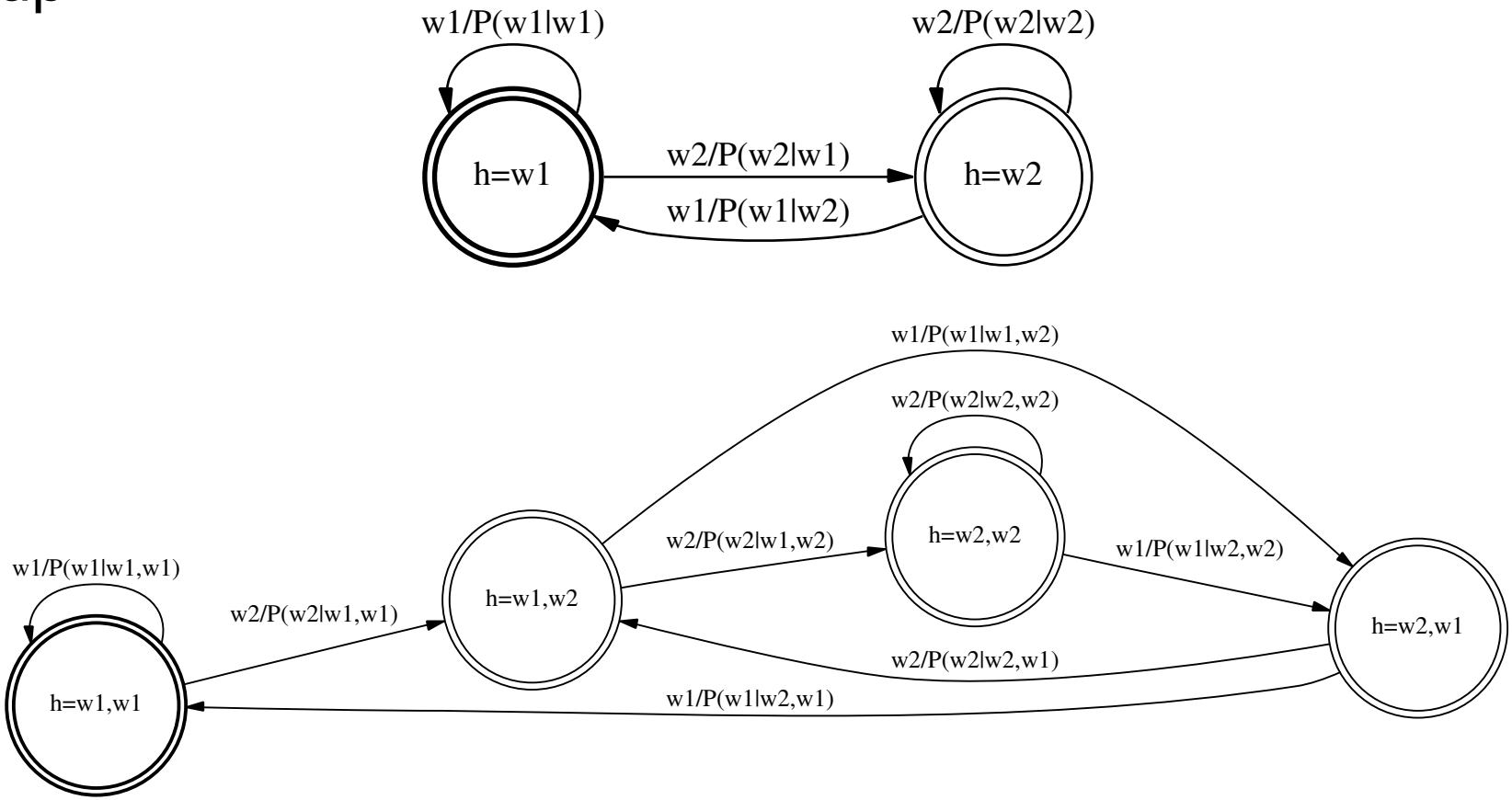


- run the Viterbi algorithm!

Decoding

Well, can we do the same thing for LVCSR?

- Issue 1: Can we express an n-gram model as an FSA?
  - yup



n-Gram Models as HMM's

- probability assigned to path is LM probability of words along that path
- do bigram example on board

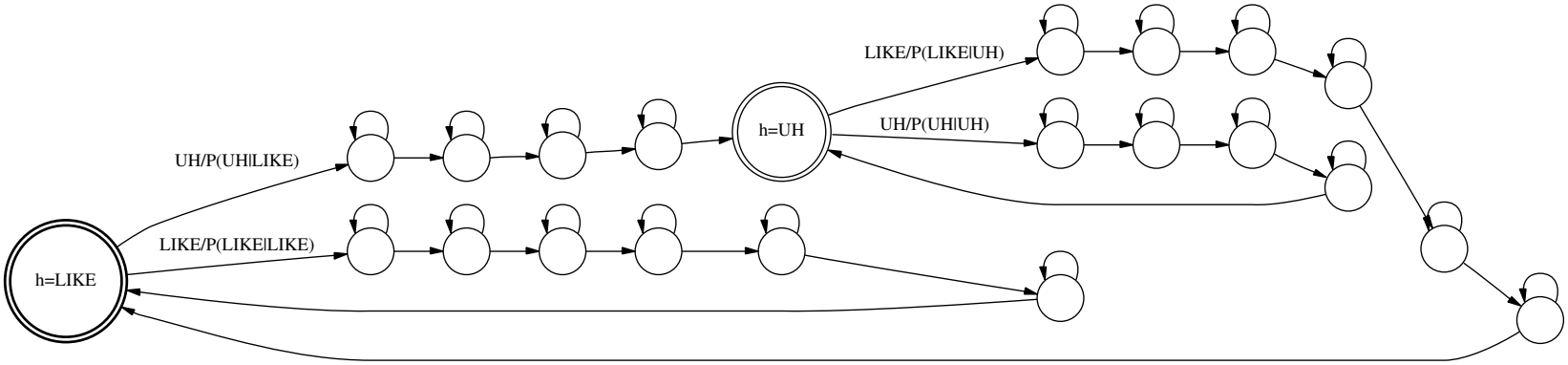
Pop Quiz

- how many states in the FSA representing an n-gram model . . .
  - with vocabulary size $|V|$?
- how many arcs?

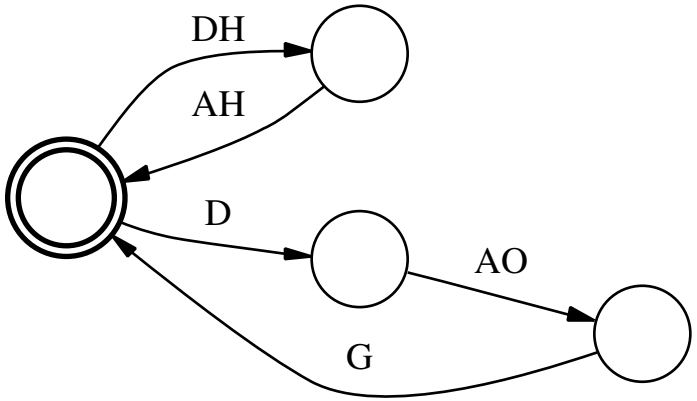
Decoding

Issue 2: How can we expand a word graph to its underlying HMM?

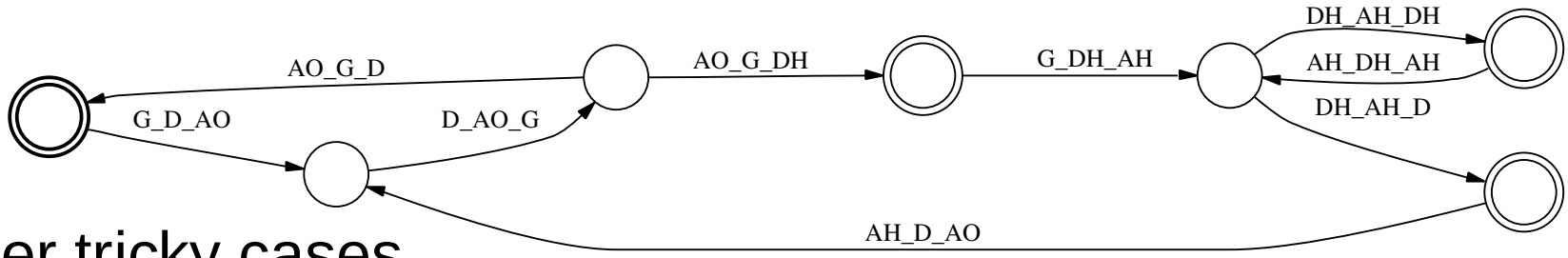
- word models
  - replace each word with its HMM
- CI phone models
  - replace each word with its phone sequence(s)
  - replace each phone with its HMM



Graph Expansion with Context-Dependent Models



- how can we do context-dependent expansion?
  - handling branch points is tricky
- example of triphone expansion

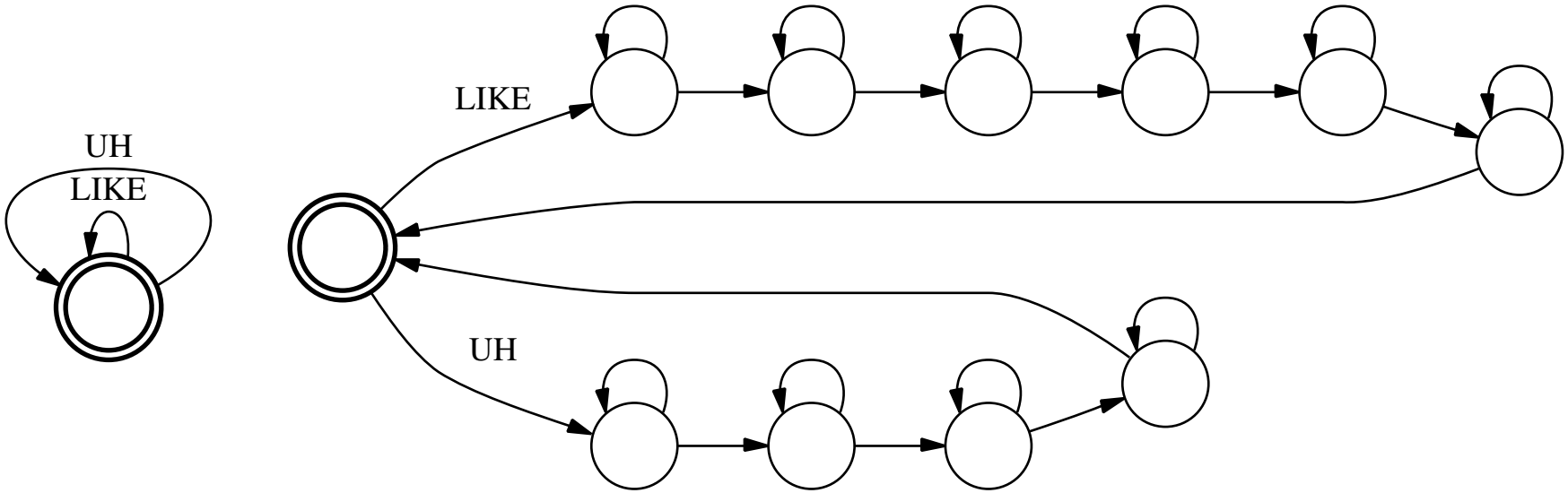


- other tricky cases
  - words consisting of a single phone
  - quinphone models

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)
- in graph expansion, just replace each word with its HMM



Graph Expansion with Context-Dependent Models

Is there a better way?

- 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)! (Unit IV)

Unit III: Decoding for LVCSR (Inefficient)

Recap

- can do same thing we do for small vocabulary decoding
  - start with LM represented as word graph
  - expand to underlying HMM
  - Viterbi
- how to do the graph expansion? FST's (Unit IV)
- how to make decoding efficient? search (Unit V)

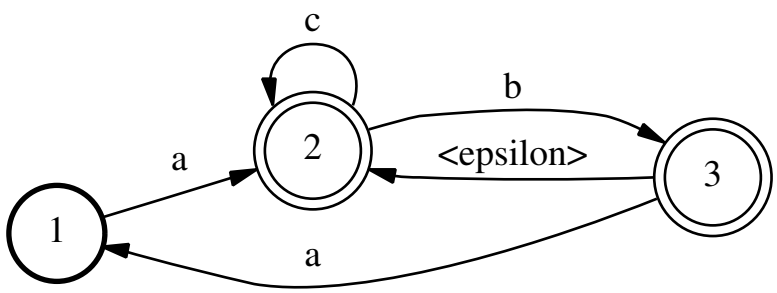
Unit IV: Introduction to Finite-State Transducers

Overview

- FST's 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

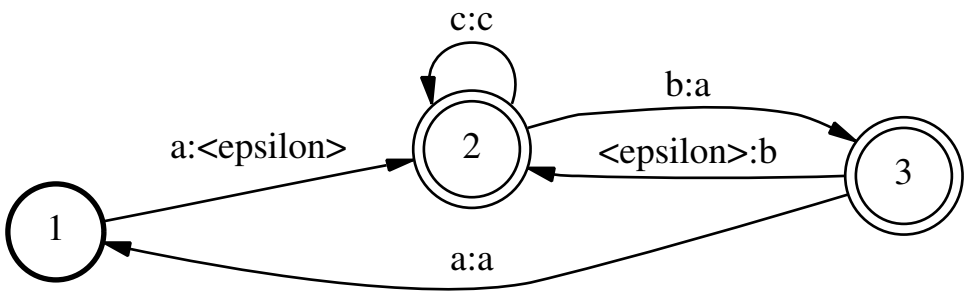


Pop Quiz

- What are the differences between the following:
  - HMM's with discrete output distributions
  - FSA's with arc probabilities
- Can they express the same class of models?

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)



Terminology

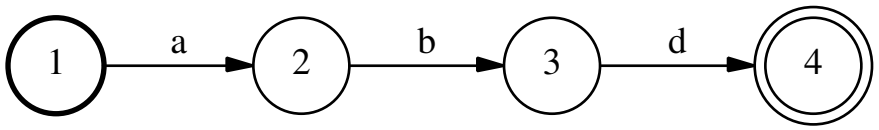
- *finite-state acceptor* (FSA): one label on each arc
- *finite-state transducer* (FST): input and output label on each arc
- *finite-state machine* (FSM): FSA or FST
  - also, *finite-state automaton*
- incidentally, an FSA can act like an FST
  - duplicate label to be both input and output label

How Can We Apply an FST to an FSA?

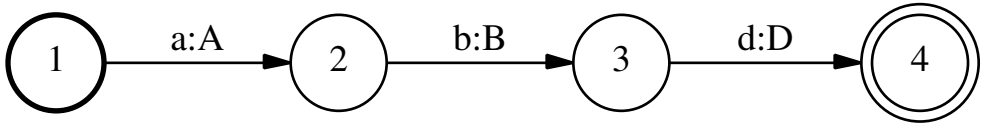
Composition operation

- perspective: rewriting/transforming token sequences

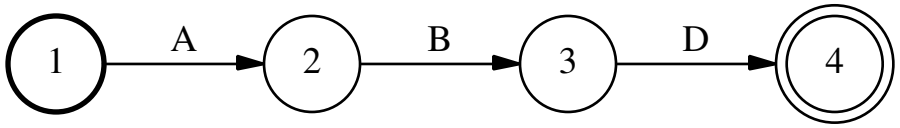
A



T



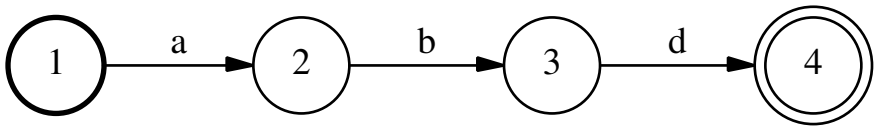
$A \circ T$



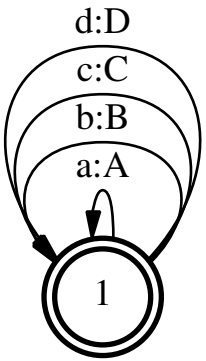
Composition

Another example

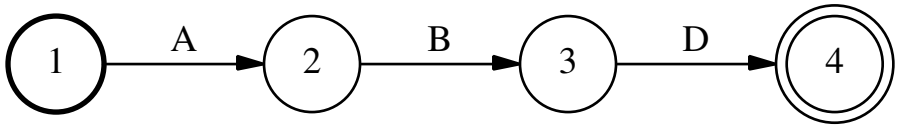
A



T



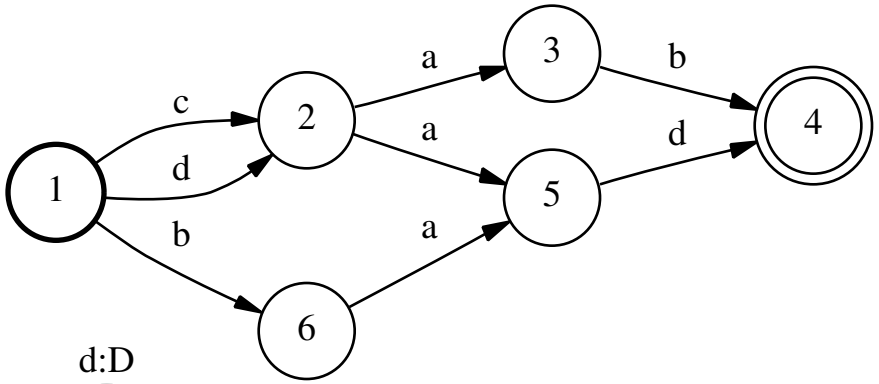
$A \circ T$



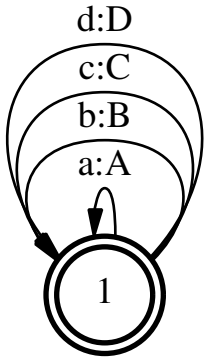
Composition

Rewriting many paths at once

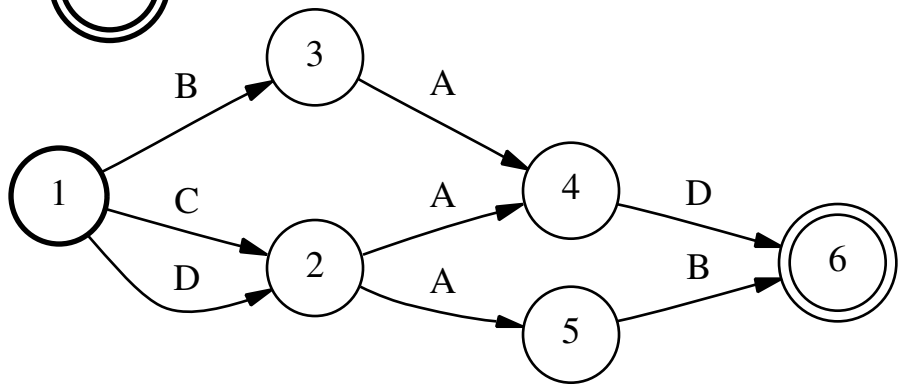
A



T



$A \circ T$



Composition

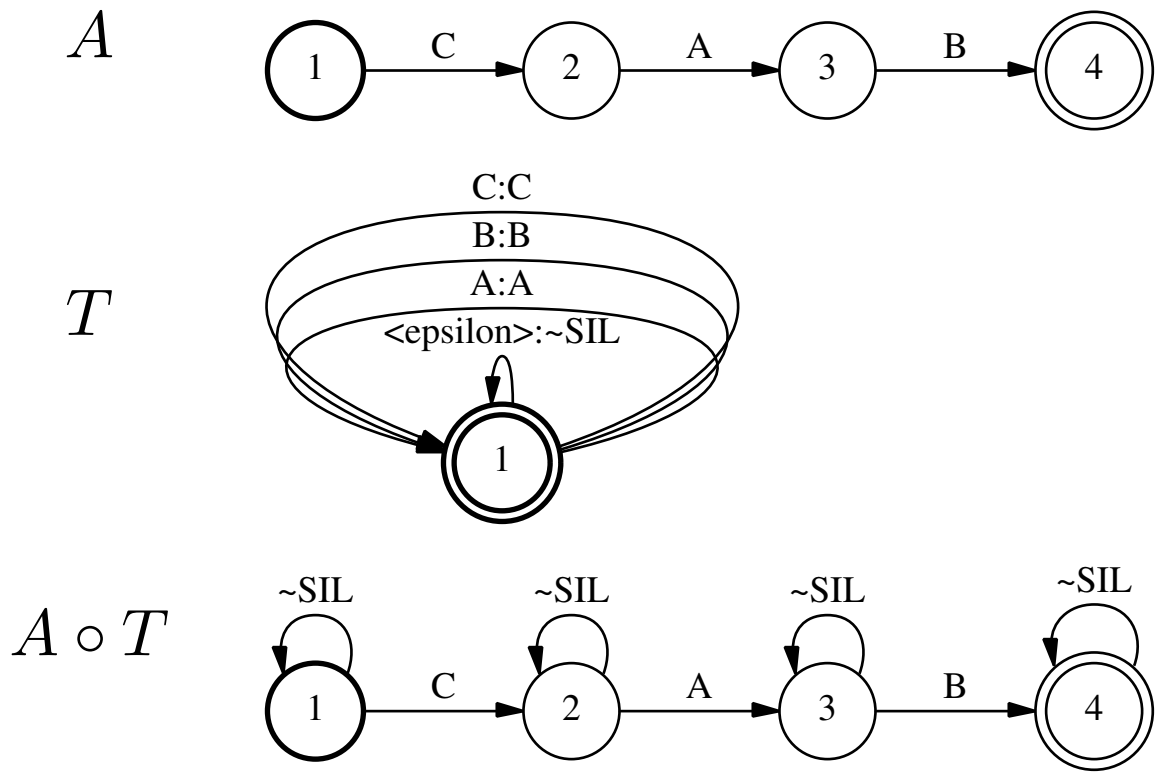
Formally, if composing FSA A with FST T to get FSA $A \circ T$:

- for every complete path (from initial to final state) in A ...
  - with input labels $i_1 \cdots i_N$ (ignoring ϵ labels) ...
- and for every complete path in T ...
  - with input labels $i_1 \cdots i_N$ and ...
  - with output labels $o_1 \cdots o_M$ ...
- there is a complete path in $A \circ T$ ...
  - with input labels $o_1 \cdots o_M$ (ignoring ϵ labels)
- we will discuss how to construct $A \circ T$ shortly

Composition

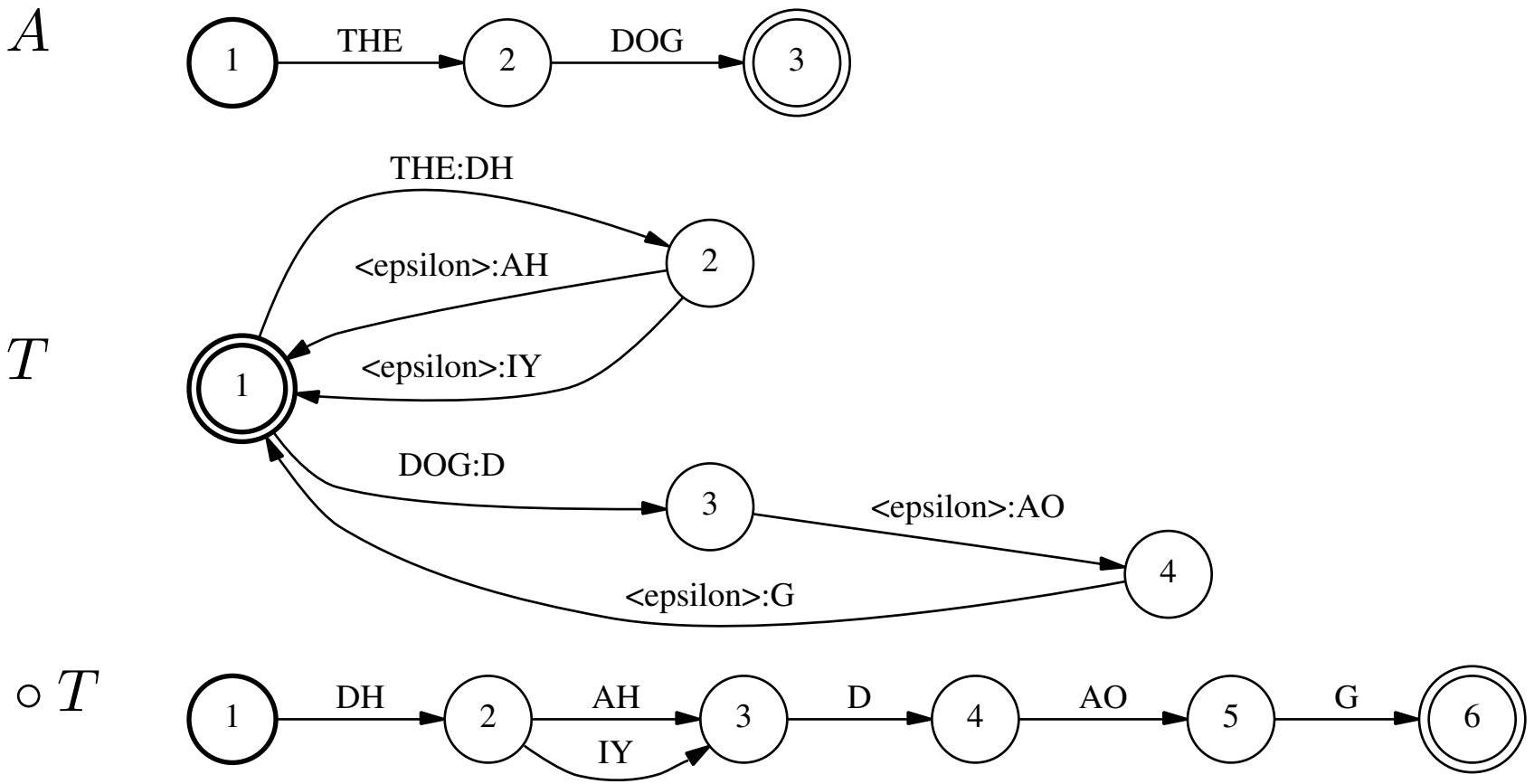
Many graph expansion operations can be represented as FST's

- example 1: optional silence insertion in training graphs



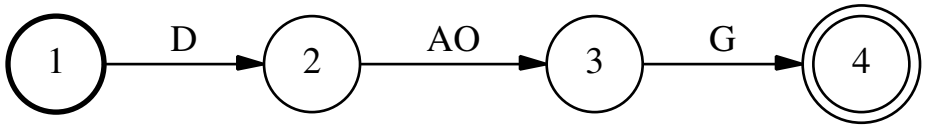
Example 2: Rewriting Words as Phone Sequences

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

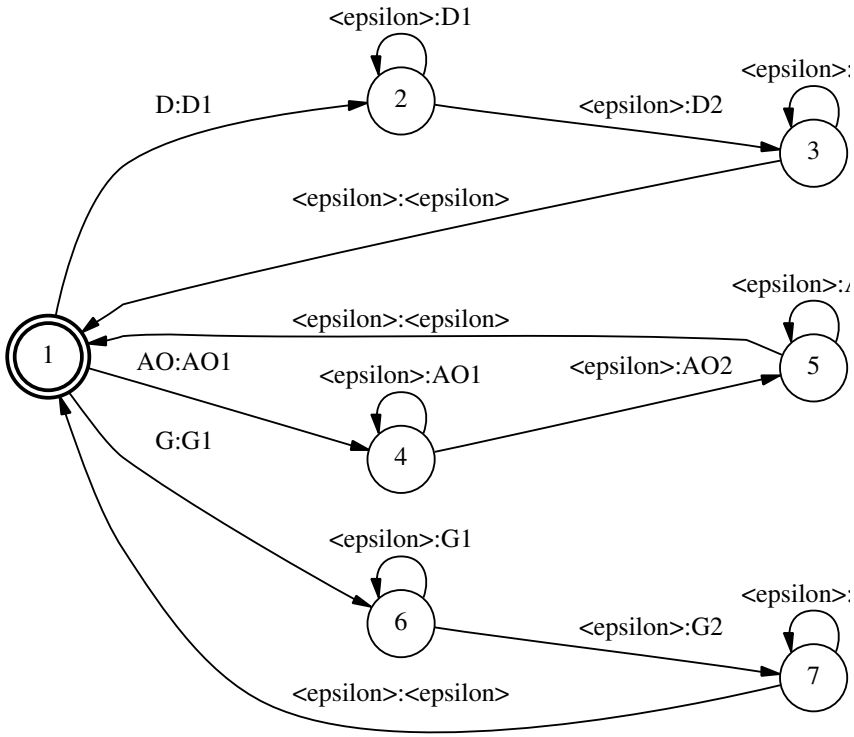


Example 3: Rewriting CI Phones as HMM's

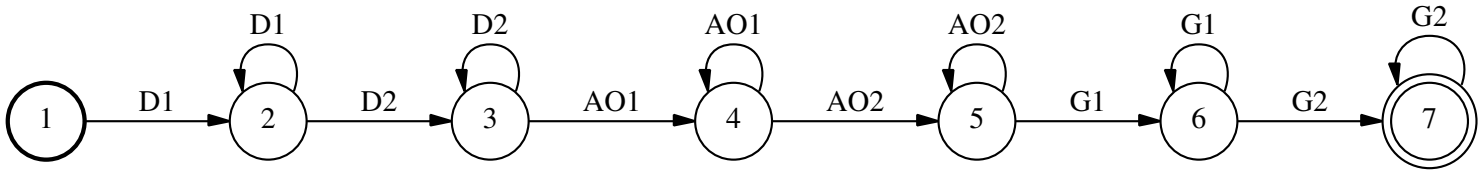
A



T



$A \circ T$

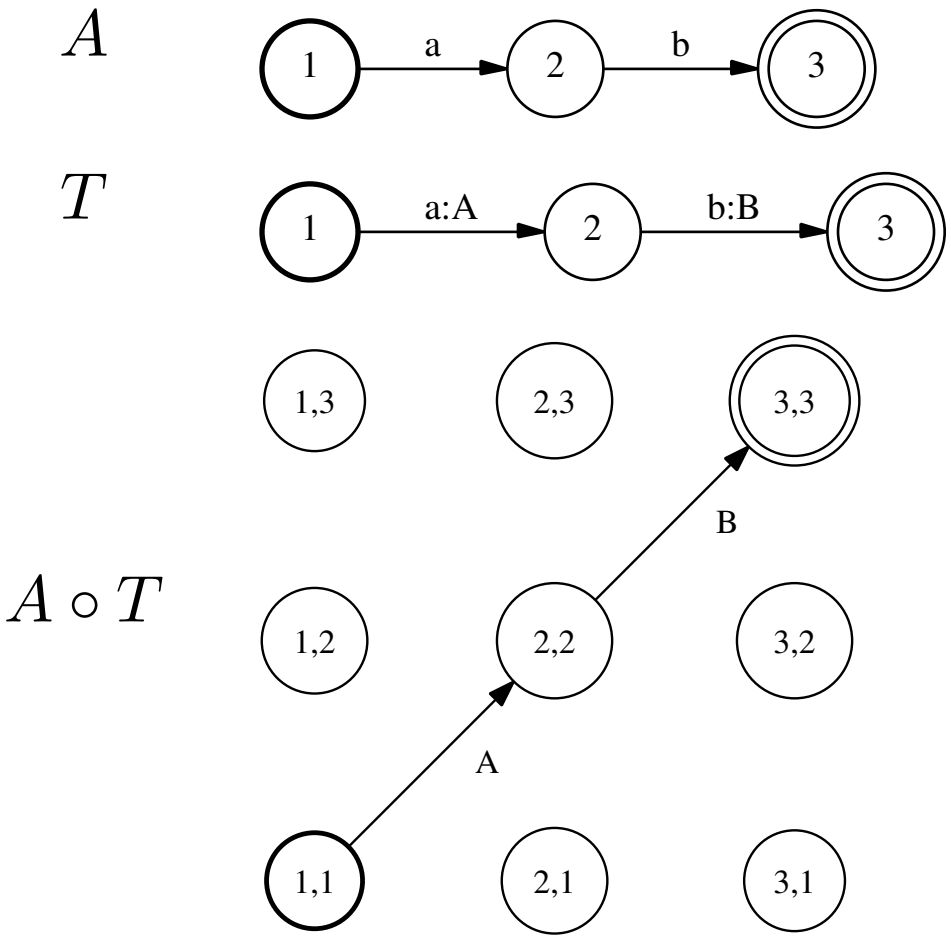


Computing Composition

- 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)
- efficient

Computing Composition

Example

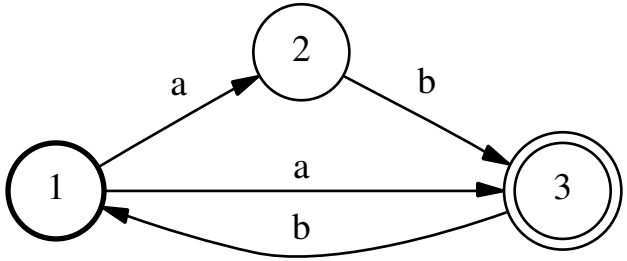


- optimization: start from initial state, build outward

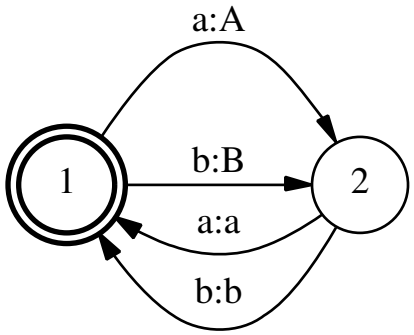
Computing Composition

Another example (see board)

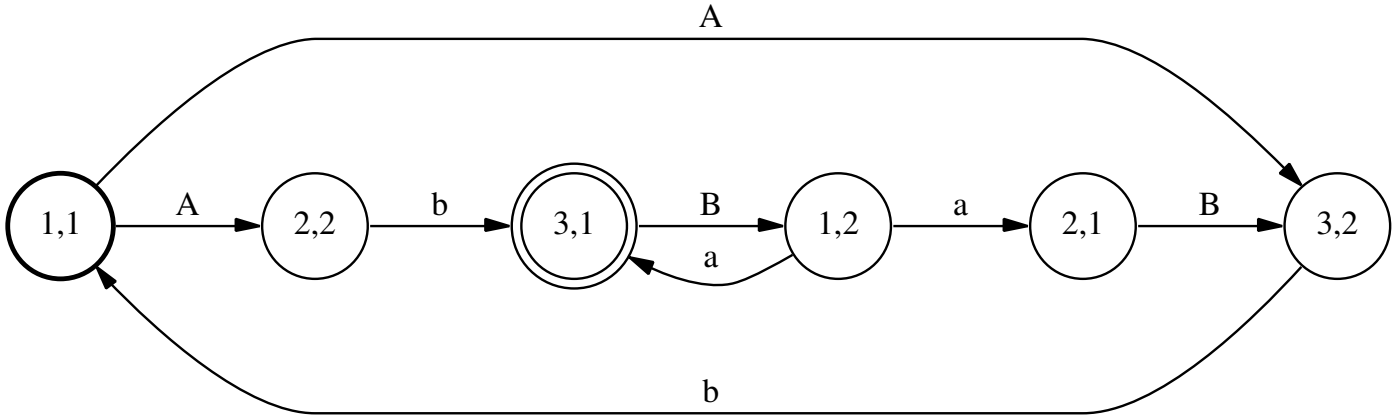
A



T



$A \circ T$

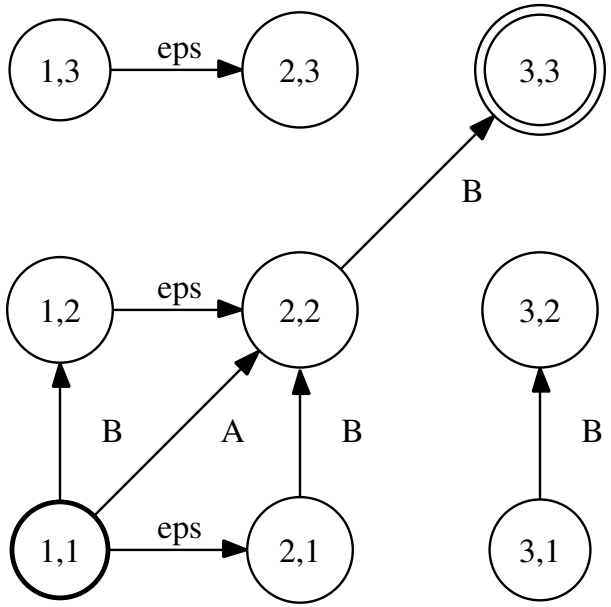


Composition and ϵ-Transitions

- 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

$A \circ T$

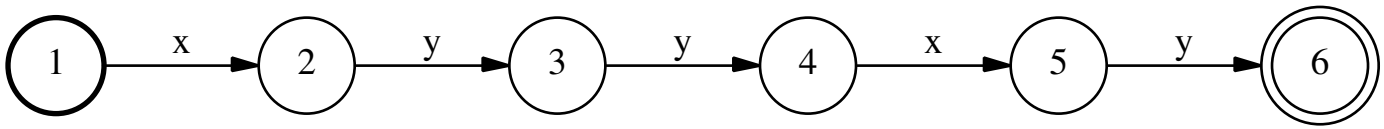


How to Express CD Expansion via FST's?

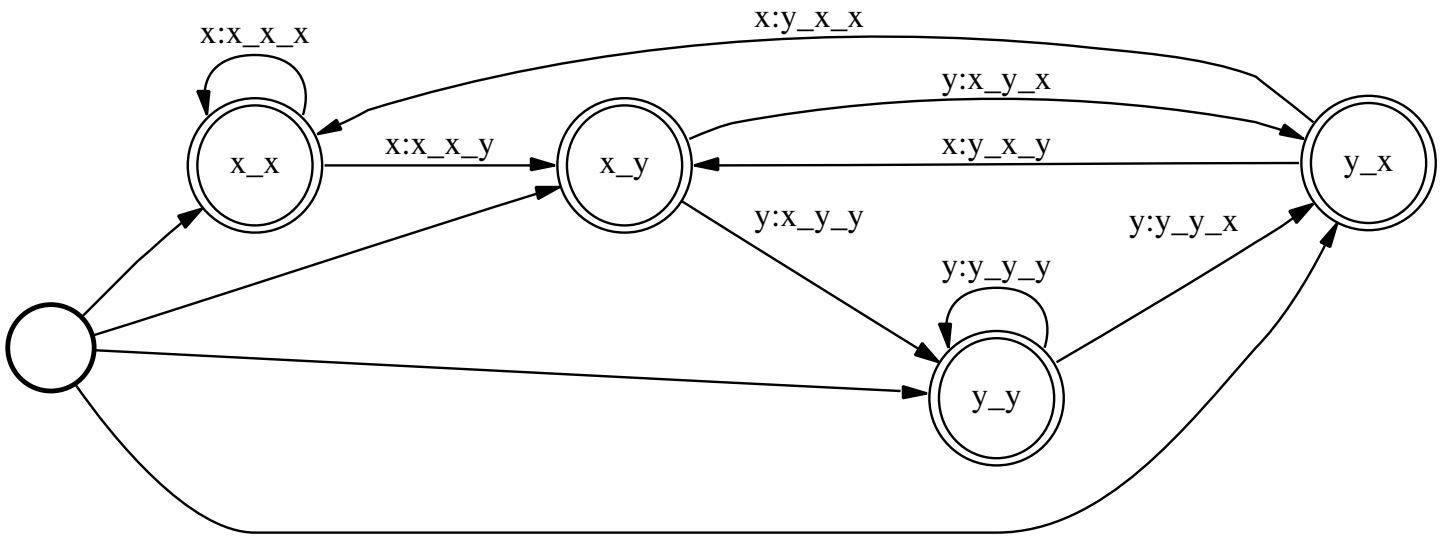
- step 1: rewrite each phone as a triphone
  - rewrite AX as DH_AX_R if DH to left, R to right
- step 2: rewrite each triphone with correct context-dependent HMM for center phone
  - just like rewriting a CI phone as its HMM
  - need to precompute HMM for each possible triphone ($\sim 50^3$)
  - example on board: CI phones $\Rightarrow$ CD phones $\Rightarrow$ HMM's

How to Express CD Expansion via FST's?

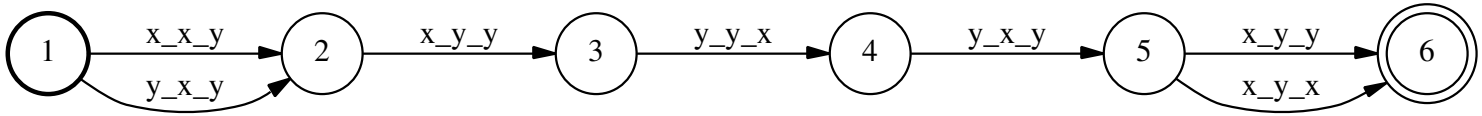
A



T

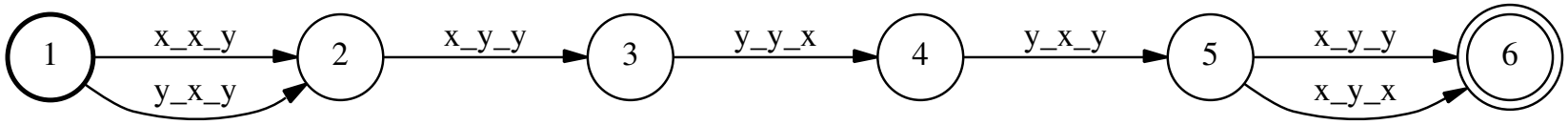


$A \circ T$



How to Express CD Expansion via FST's?

Example



- 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)

Unit IV: Introduction to Finite-State Transducers

What we've learned so far:

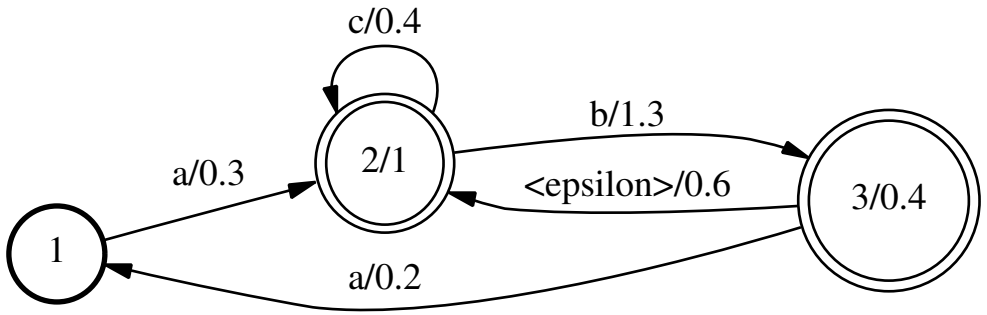
- graph expansion can be expressed as series of composition operations
  - need to build FST to represent each expansion step, *e.g.*,

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

  - with composition operation, we're done!
- composition is efficient
- context-dependent expansion can be handled effortlessly

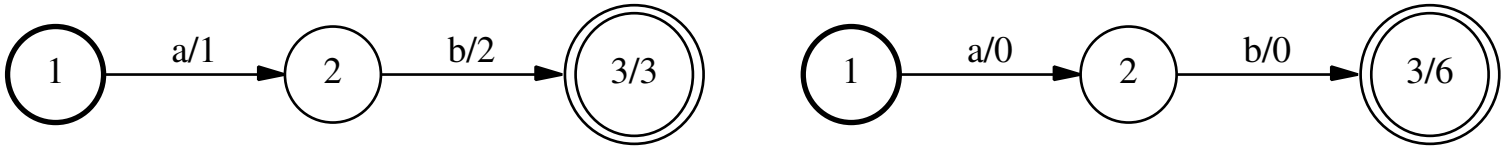
What About Those Probability Thingies?

- *e.g.*, to hold language model probs, transition probs, etc.
- FSM's $\Rightarrow$ *weighted* FSM's
  - WFSA's, WFST's
- each arc has a score or *cost*
  - so do final states

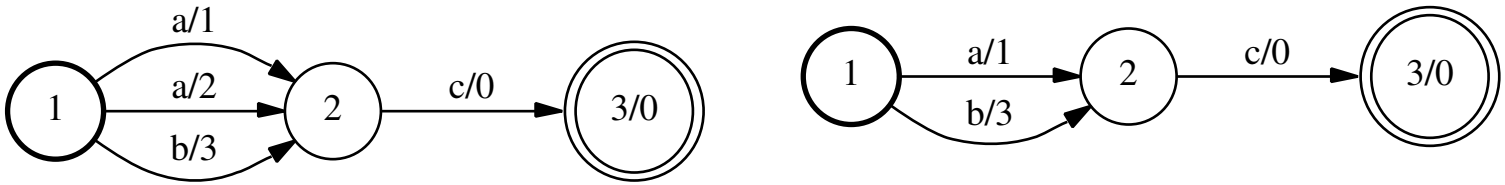


How Are Arc Costs and Probabilities Related?

- 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



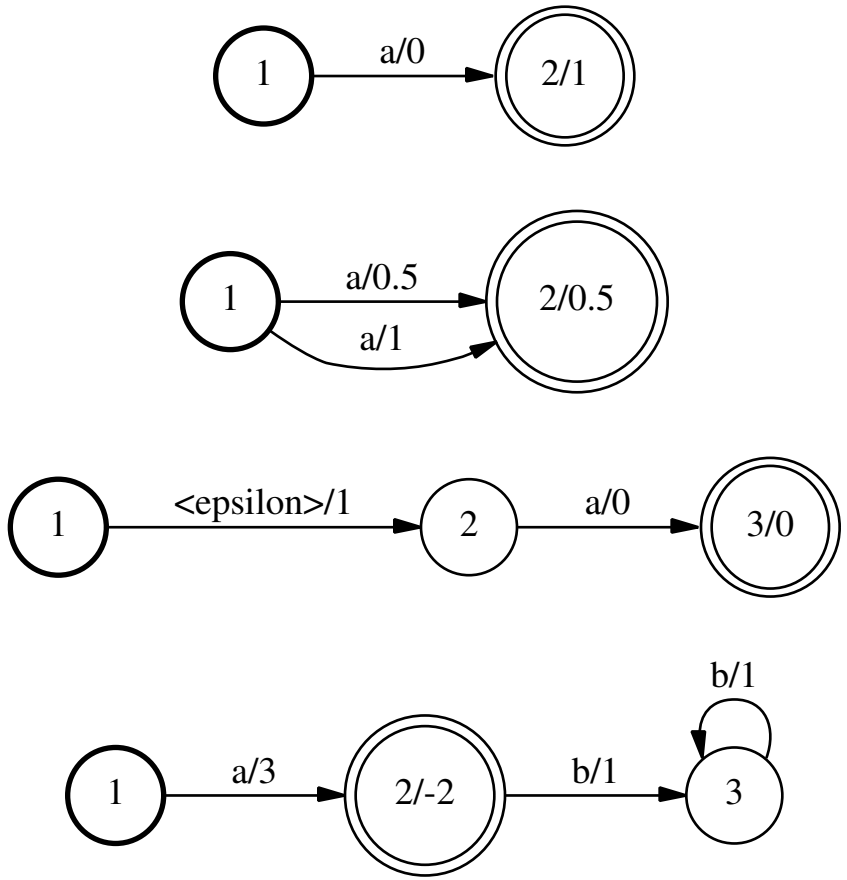
- if two paths have same labels, can be combined into one
  - typically, use $\min$ operator to compute new cost



- operations $(+, \min)$ form a *semiring* (the *tropical* semiring)
  - other semirings are possible

Which Of These Is Different From the Others?

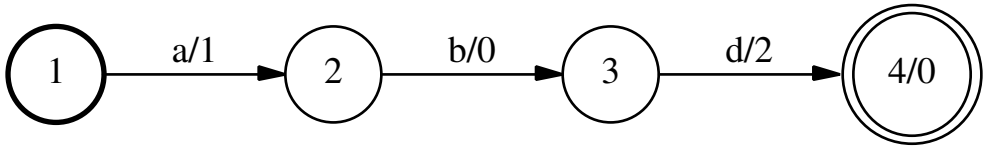
- FSM's are equivalent if same label sequences with same costs



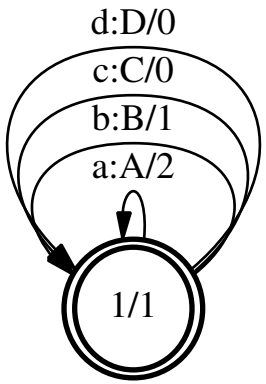
Weighted Composition

Just add arc costs

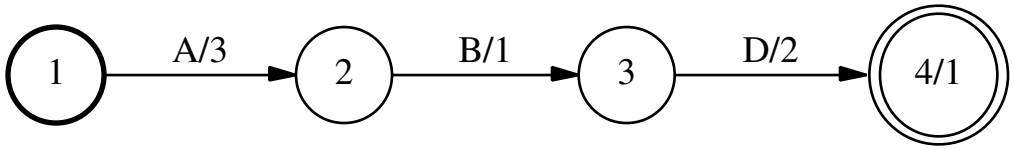
A



T



$A \circ T$



Weighted Graph Expansion

- start with weighted FSA representing language model
- use composition to apply FST for each level of expansion
  - scores/logprobs will be accumulated
  - logprobs may move around along paths
  - all that matters for Viterbi is total score of paths

Unit IV: Introduction to Finite-State Transducers

Recap

- WFSA's and WFST's can represent many important structures in ASR
- composition can do lots of useful things, including . . .
  - transforming arc labels
  - context-dependent expansion
  - adding in new arc scores
  - restricting the set of allowed paths

Road Map

Where are we going?

- Unit I: you do not talk about Unit I
- Unit II: acoustic model training for LVCSR
- Unit III: decoding for LVCSR (inefficient)
  - Unit IV: introduction to finite-state transducers
- Unit 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)?