Lecture 5

The Big Picture/Language Modeling

Michael Picheny, Bhuvana Ramabhadran, Stanley F. Chen, Markus Nussbaum-Thom

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

17 February 2016

Administrivia

- Slides posted before lecture may not match lecture.
- Lab 1
  - Not graded yet; will be graded by next lecture?
  - Awards ceremony for evaluation next week.
  - Grading: what's up with the optional exercises?
- Lab 2
  - Due nine days from now (Friday, Feb. 26) at 6pm.
  - Start early! Avail yourself of Piazza.

Feedback

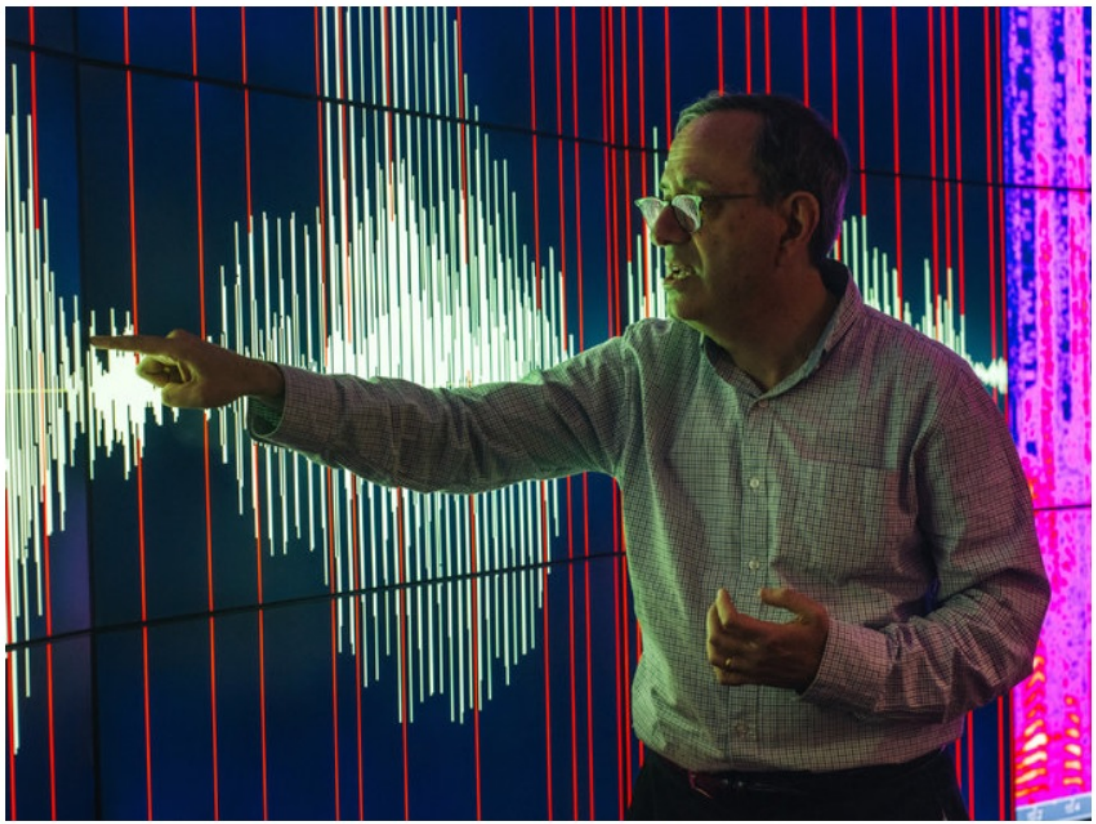
- Clear (4); mostly clear (2); unclear (3).
- Pace: fast (3); OK (2).
- Muddiest: HMM's in general (1); Viterbi (1); FB (1).
- Comments (2+ votes):
  - want better/clearer examples (5)
  - spend more time walking through examples (3)
  - spend more time on high-level intuition before getting into details (3)
  - good examples (2)

Celebrity Sighting

TECHNOLOGY

Creating a Computer Voice That People Like

By JOHN MARKOFF FEB. 14, 2016



Michael Picheny, a senior manager at the Watson Multimodal Lab for IBM Research.

Cole Wilson for The New York Times

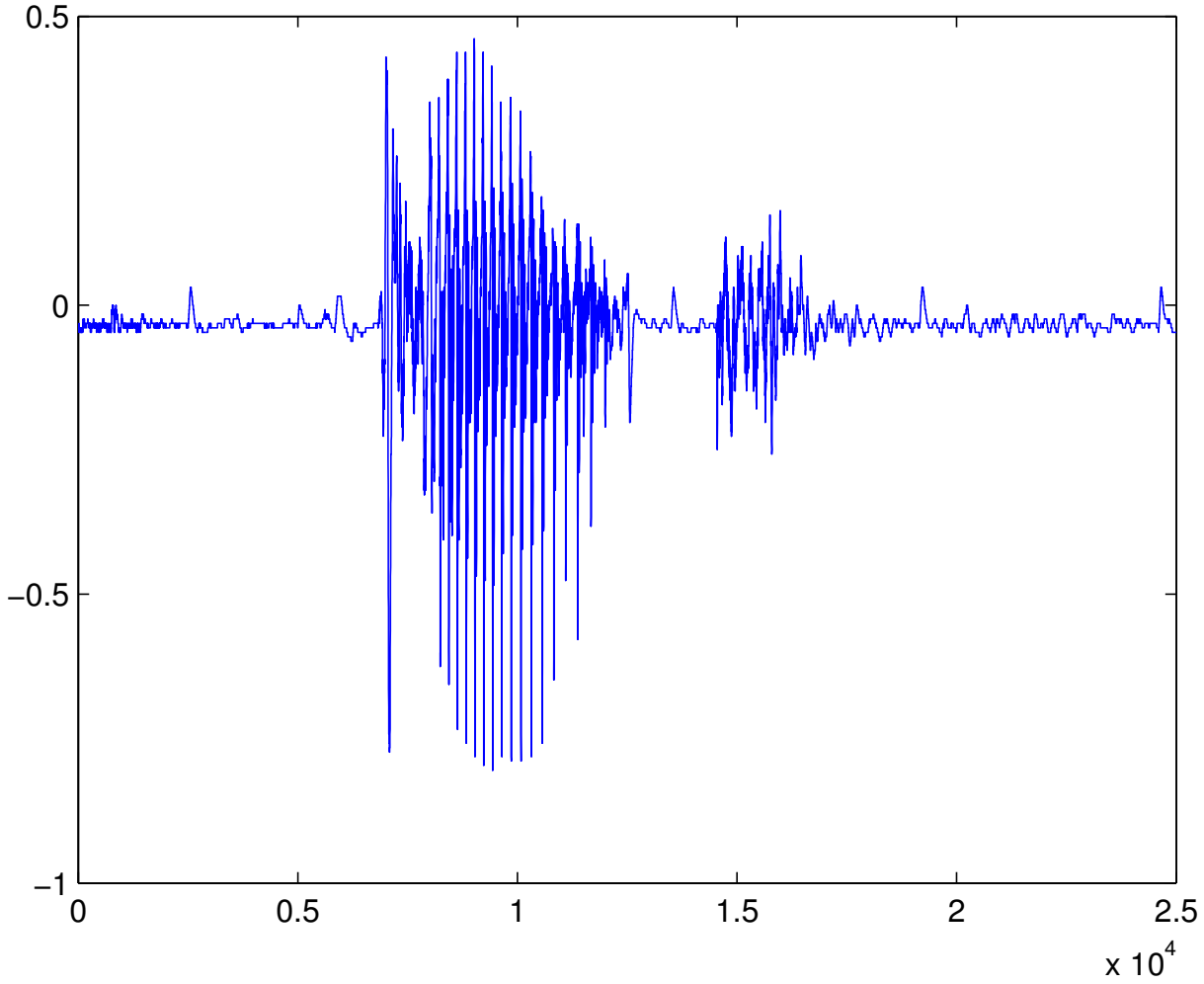
New York Times

Part I

The HMM/GMM Framework

Where Are We?

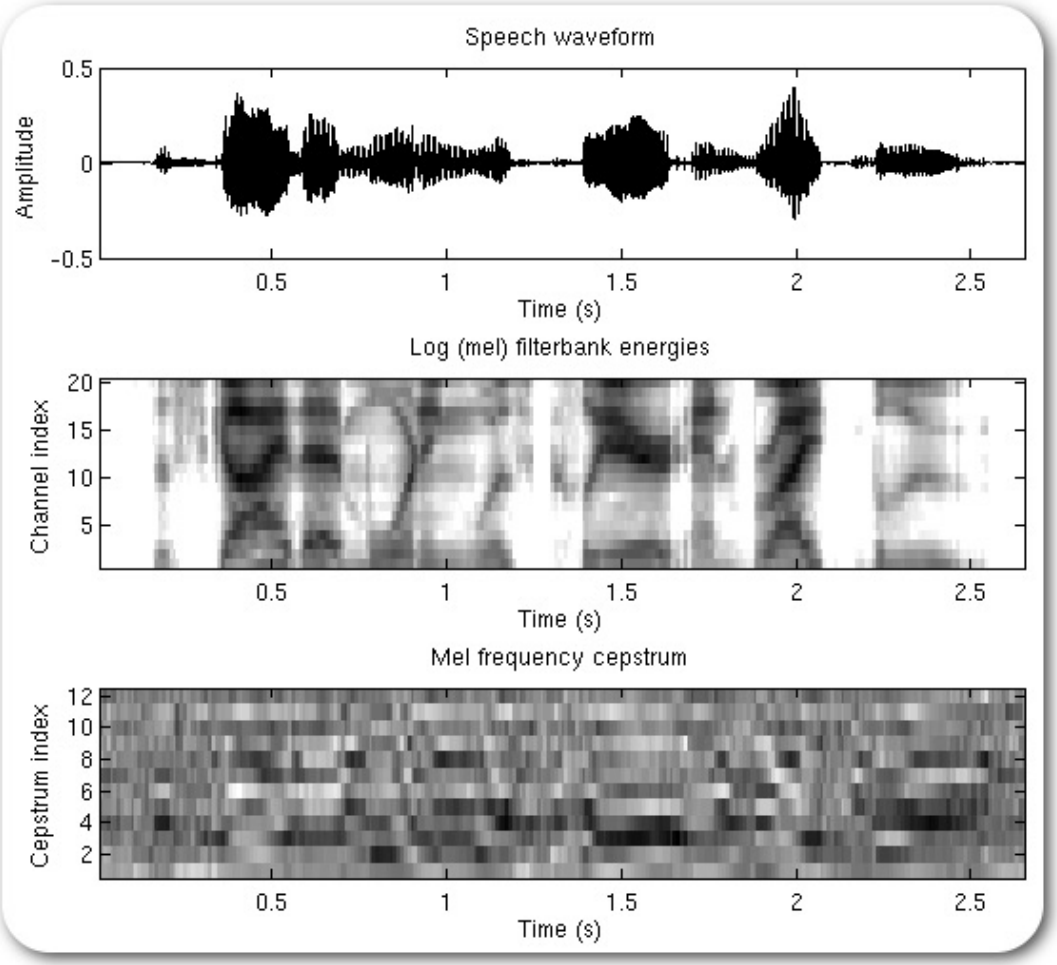
The Raw Data



- What do we do with waveforms?

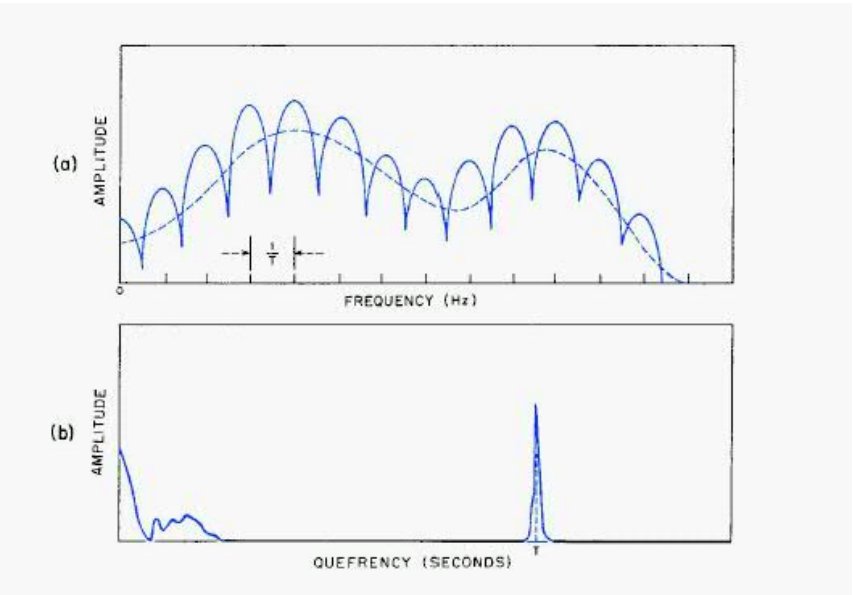
Front End Processing

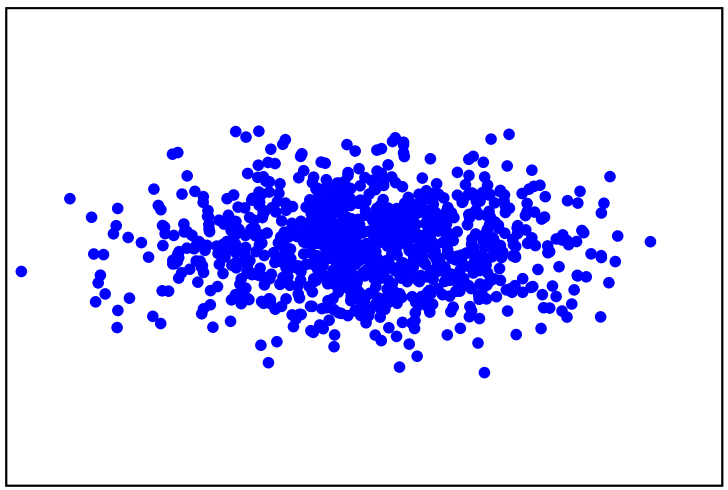
- Convert waveform to *features*.



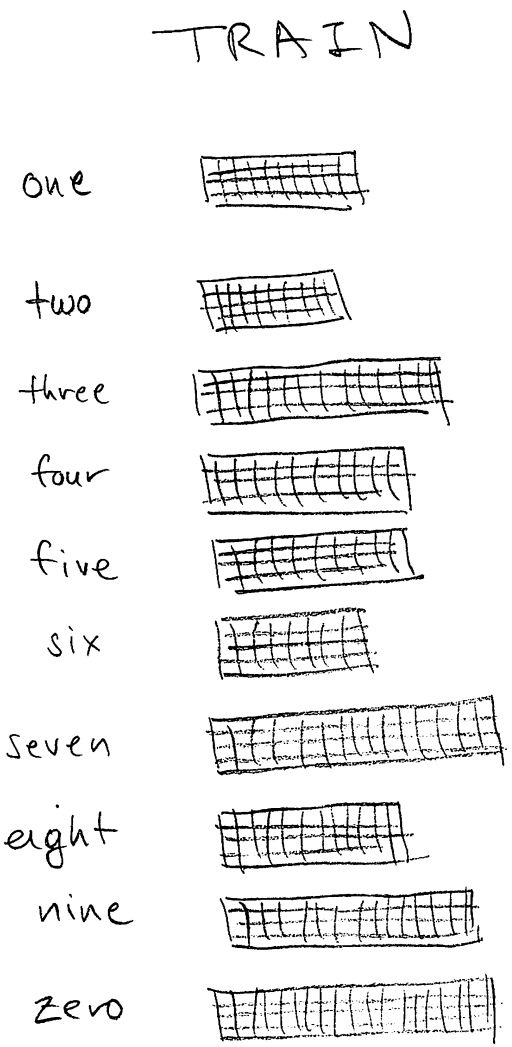
What Have We Gained?

- Time domain $\Rightarrow$ frequency domain.
- Removed vocal-fold excitation.
- Made features independent.



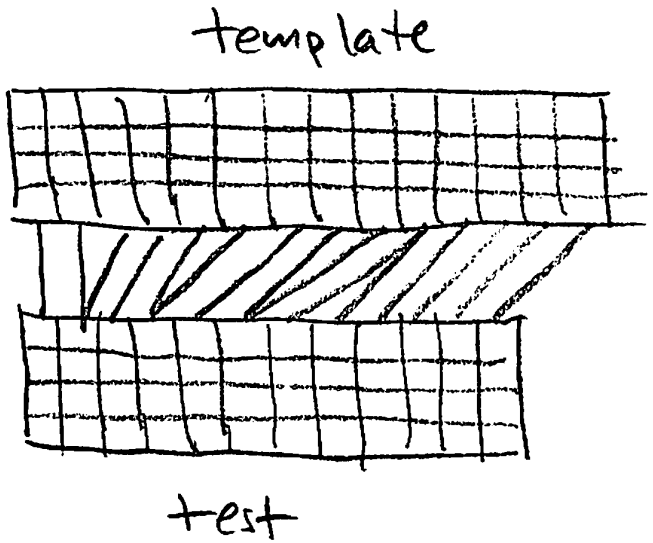


ASR 1.0: Dynamic Time Warping

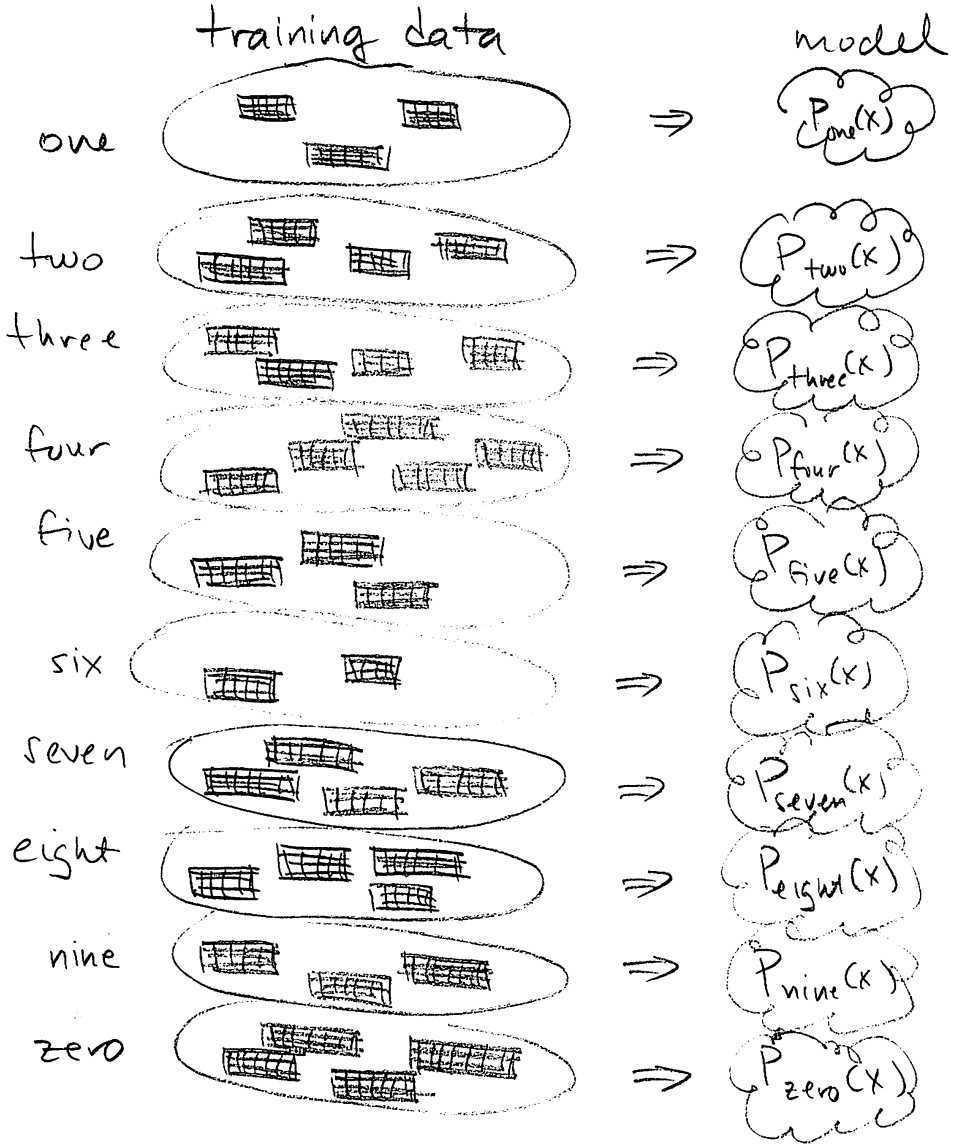


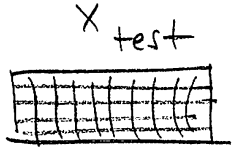
Computing the Distance Between Utterances

- Find “best” alignment between frames.
- Sum distances between aligned frames.
- Sum penalties for “weird” alignments.

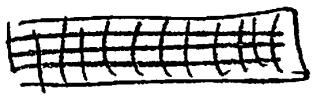
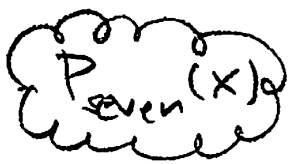


ASR 2.0: The HMM/GMM Framework





Notation

	notation	picture
feature vector	x	
word	ω	seven
model	$P_\omega(x)$	

How Do We Do Recognition?

- $\mathbf{x}_{\text{test}}$ = test features; $P_\omega(\mathbf{x})$ = word model.

$$\text{(answer)} = ???$$

$$\text{(answer)} = \arg\max_{\omega \in \text{vocab}} P_\omega(\mathbf{x}_{\text{test}})$$

- Return the word whose model . . .
- Assigns the highest prob to the utterance.

Putting it All Together

- $P_\omega(\mathbf{x}) = ???$
- How do we actually train?
- How do we actually *decode*?

Where Are We?

So What's the Model?

$$P_\omega(\mathbf{x}) = ???$$

- Frequency that word ω generates features $\mathbf{x}$.
- Has something to do with HMM's and GMM's.

A Word Is A Sequence of Sounds

- *e.g.*, the word ONE: W → AH → N.
- Phoneme inventory.

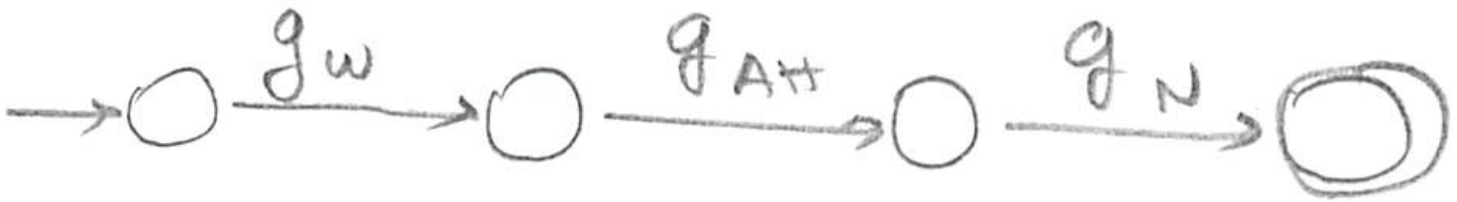
AA	AE	AH	AO	AW	AX	AXR	AY	B
BD	CH	D	DD	DH	DX	EH	ER	EY
F	G	GD	HH	IH	IX	IY	JH	K
KD	L	M	N	NG	OW	OY	P	PD
R	S	SH	T	TD	TH	TS	UH	UW
V	W	X	Y	Z	ZH			

- What sounds make up TWO?
- What do we use to model sequences?

HMM, v1.0

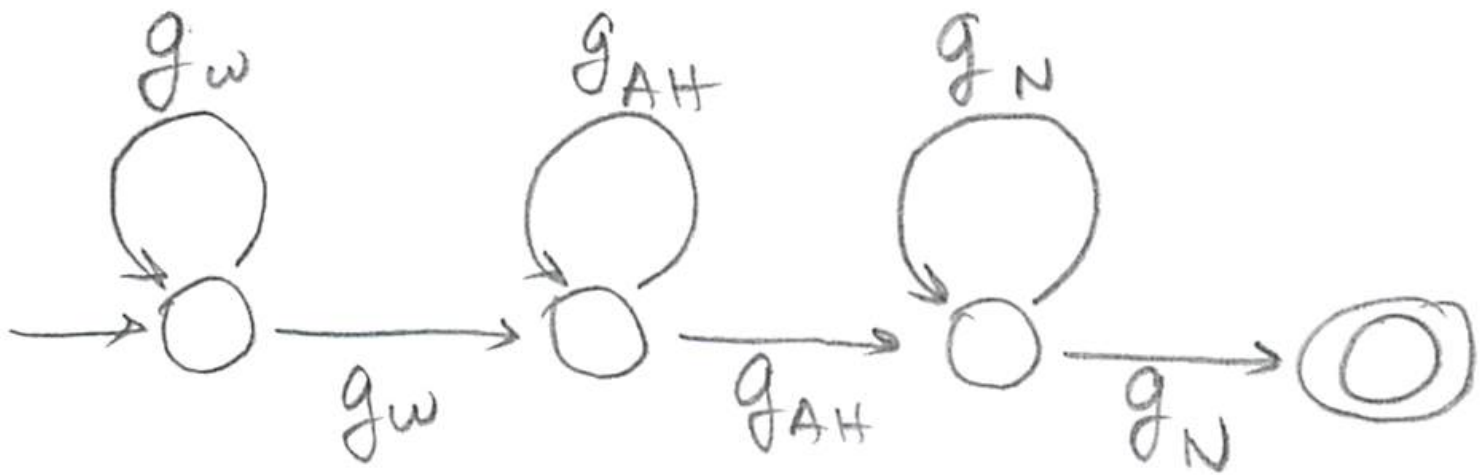
- Outputs on arcs, not states.
- What's the problem? What are the outputs?

HMM, v2.0



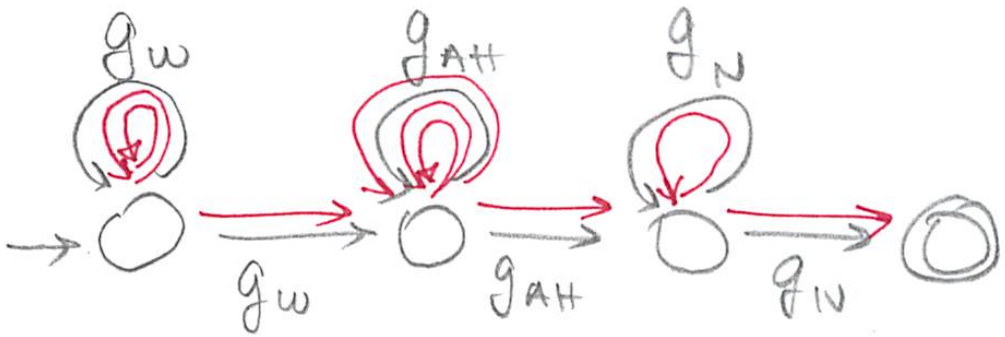
- What's the problem? How many frames per phoneme?

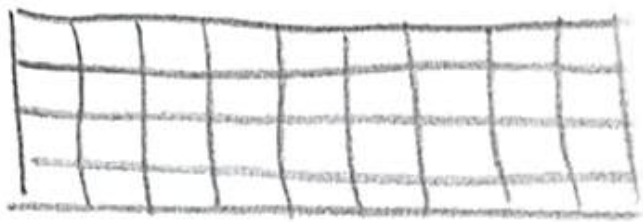
HMM, v3.0



- Are we done?

Concept: Alignment $\Leftrightarrow$ Path





- Path through HMM $\Rightarrow$ sequence of arcs, one per frame.
- Notation: $A = a_1 \cdots a_T$.
- a_t = which arc generated frame t.

The Game Plan

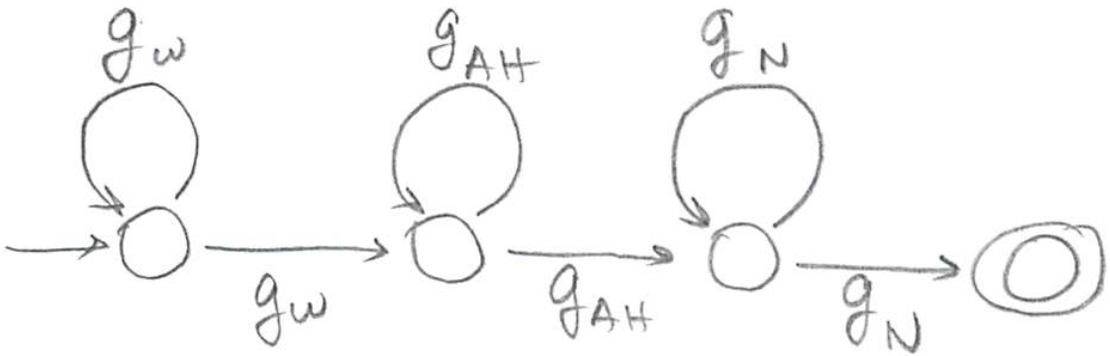
- Express $P_\omega(\mathbf{x})$, the total prob of $\mathbf{x}$ ...
- In terms of $P_\omega(\mathbf{x}, A)$, the prob of a single path.
- How?

$$P(\mathbf{x}) = \sum_{\text{paths } A} (\text{path prob})$$
$$= \sum_{\text{paths } A} P(\mathbf{x}, A)$$

- Sum over all paths.

How To Compute the Likelihood of a Path?

- Path: $A = a_1 \cdots a_T$.



$$P(\mathbf{x}, A) = \prod_{t=1}^{T} (\text{arc prob}) \times (\text{output prob})$$

$$= \prod_{t=1}^{T} p_{a_t} \times P(\vec{x}_t | a_t)$$

- Multiply arc, output probs along path.

What Do Output Probabilities Look Like?

- Mixture of diagonal-covariance Gaussians.

$$P(\vec{x}|a) = \sum_{\text{comp } j} (\text{mixture wgt}) \prod_{\text{dim } d} (\text{Gaussian for dim } d)$$

$$= \sum_{\text{comp } j} p_{a,j} \prod_{\text{dim } d} \mathcal{N}(x_d; \mu_{a,j,d}, \sigma_{a,j,d})$$

The Full Model

$$P(\mathbf{x}) = \sum_{\text{paths } A} P(\mathbf{x}, A)$$

$$= \sum_{\text{paths } A} \prod_{t=1}^{T} p_{a_t} \times P(\vec{x}_t | a_t)$$

$$= \sum_{\text{paths } A} \prod_{t=1}^{T} p_{a_t} \sum_{\text{comp } j} p_{a_t,j} \prod_{\text{dim } d} \mathcal{N}(x_{t,d}; \mu_{a_t,j,d}, \sigma^2_{a_t,j,d})$$

- p_a — transition probability for arc a.
- $p_{a,j}$ — mixture weight, jth component of GMM on arc a.
- $\mu_{a,j,d}$ — mean, dth dim, jth component, GMM on arc a.
- $\sigma^2_{a,j,d}$ — variance, dth dim, jth component, GMM on arc a.

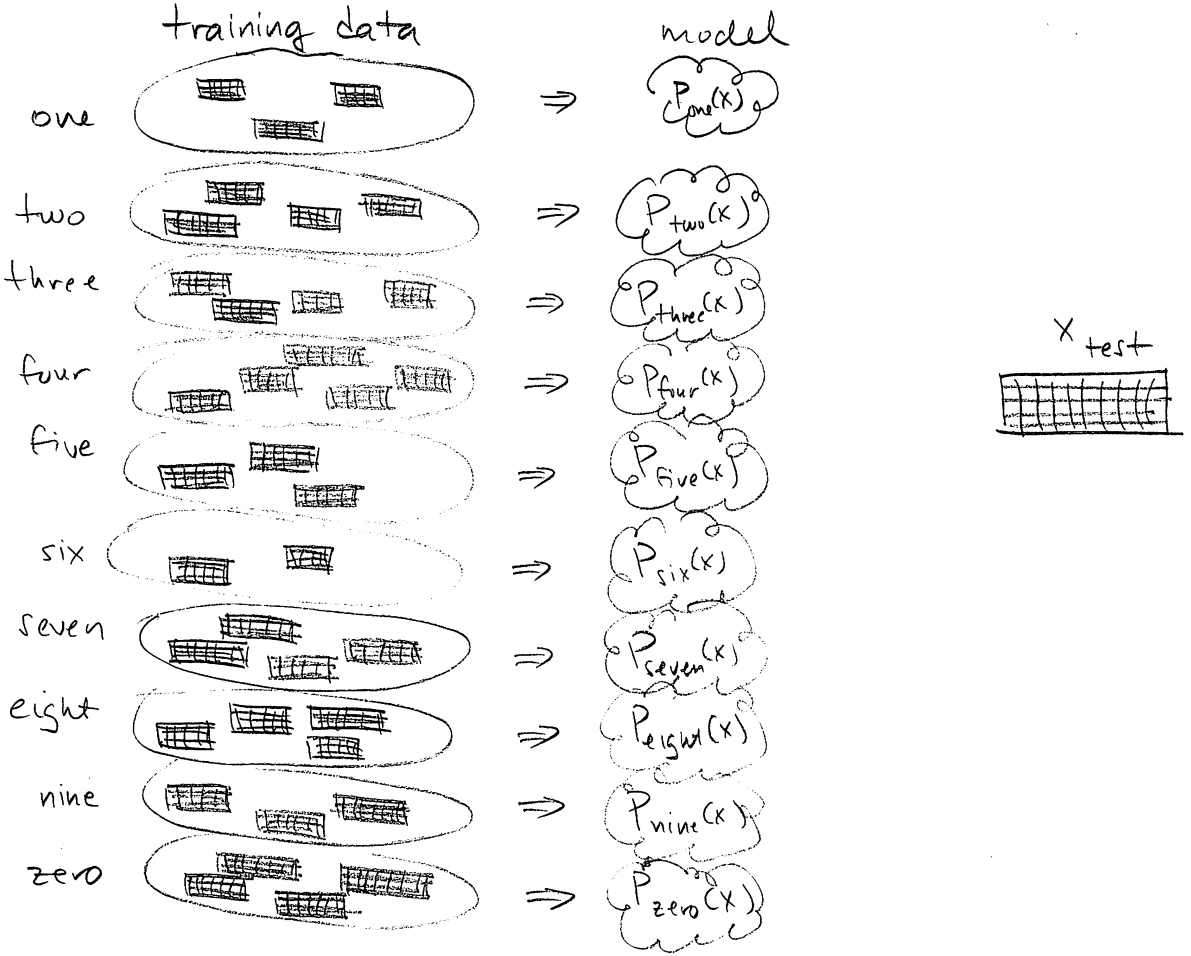
Pop Quiz

- What was the equation on the last slide?

Where Are We?

Training

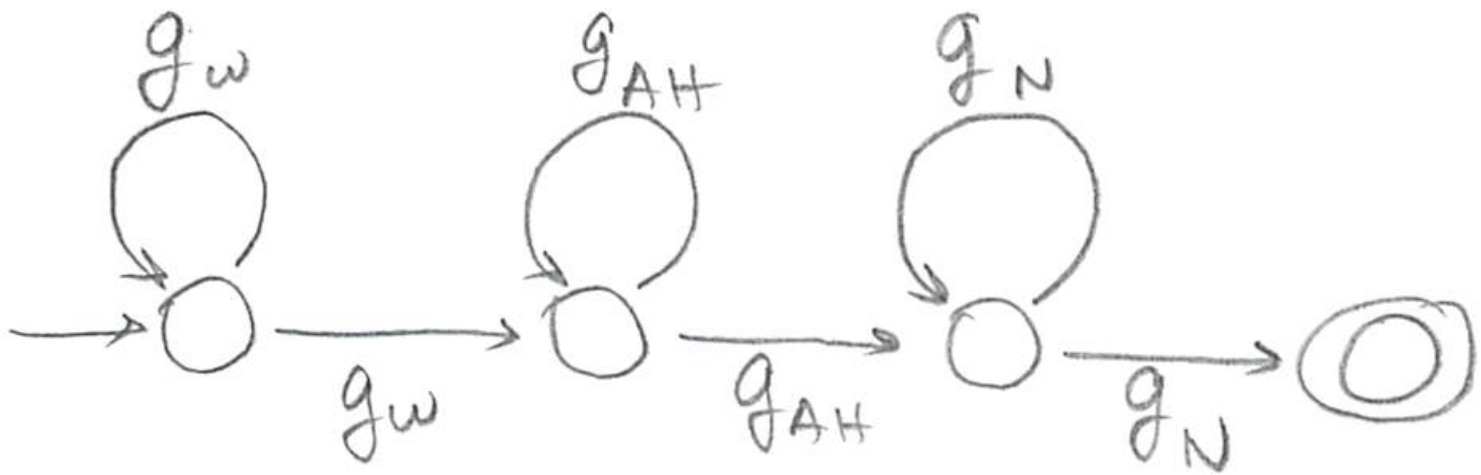
- How to create model $P_\omega(\mathbf{x})$ from examples $\mathbf{x}_{\omega,1}$, $\mathbf{x}_{\omega,2}$, ...?



What is the Goal of Training?

- To estimate *parameters* . . .
- To maximize likelihood of training data.

What Are the Model Parameters?



- p_a — transition probability for arc a.
- $p_{a,j}$ — mixture weight, jth component of GMM on arc a.
- $\mu_{a,j,d}$ — mean, dth dim, jth component, GMM on arc a.
- $\sigma^2_{a,j,d}$ — variance, dth dim, jth component, GMM on arc a.

Warm-Up: Non-Hidden ML Estimation

- *e.g.*, Gaussian estimation, non-hidden Markov Models.
- How to do this? (Hint: ??? and ???.)

parameter	description	statistic
p_a	arc prob	# times arc taken
$p_{a,j}$	mixture wgt	# times component used
$\mu_{a,j,d}$	mean	x_d
$\sigma^2_{a,j,d}$	variance	x_d^2

- Count and normalize.
  - *i.e.*, collect a statistic; divide by normalizer count.

How To Estimate Hidden Models?

- The EM algorithm $\Rightarrow$ FB algorithm for HMM's.
- Hill-climbing maximum-likelihood estimation.

The EM Algorithm

- Expectation step.
  - Using current model, compute posterior counts . . .
  - Prob that thing occurred at time t.
- Maximization step.
  - Like non-hidden MLE, except . . .
  - Use fractional posterior counts instead of whole counts.
- Repeat.

E step: Calculating Posterior Counts

- *e.g.*, posterior count $\gamma(a, t)$ of taking arc a at time t.

$$\begin{aligned}
\gamma(a,t) &= \frac{P(\text{paths with arc } a \text{ at time } t)}{P(\text{all paths})} \\
&= \frac{1}{P(\mathbf{x})} \times P(\text{paths from start to src}(a)) \times \\
&\quad P(\text{arc } a \text{ at time } t) \times P(\text{paths from dst}(a) \text{ to end}) \\
&= \frac{1}{P(\mathbf{x})} \times \alpha(\text{src}(a), t-1) \times p_a \times P(\vec{x}_t | a) \times \beta(\text{dst}(a), t)
\end{aligned}$$

- Do Forward algorithm: $\alpha(S, t)$, $P(\mathbf{x})$.
- Do Backward algorithm: $\beta(S, t)$.
- Read off posterior counts.

M step: Non-Hidden ML Estimation

- Count and normalize.
- Same stats as non-hidden, except normalizer is fractional.
- *e.g.*, arc prob p_a

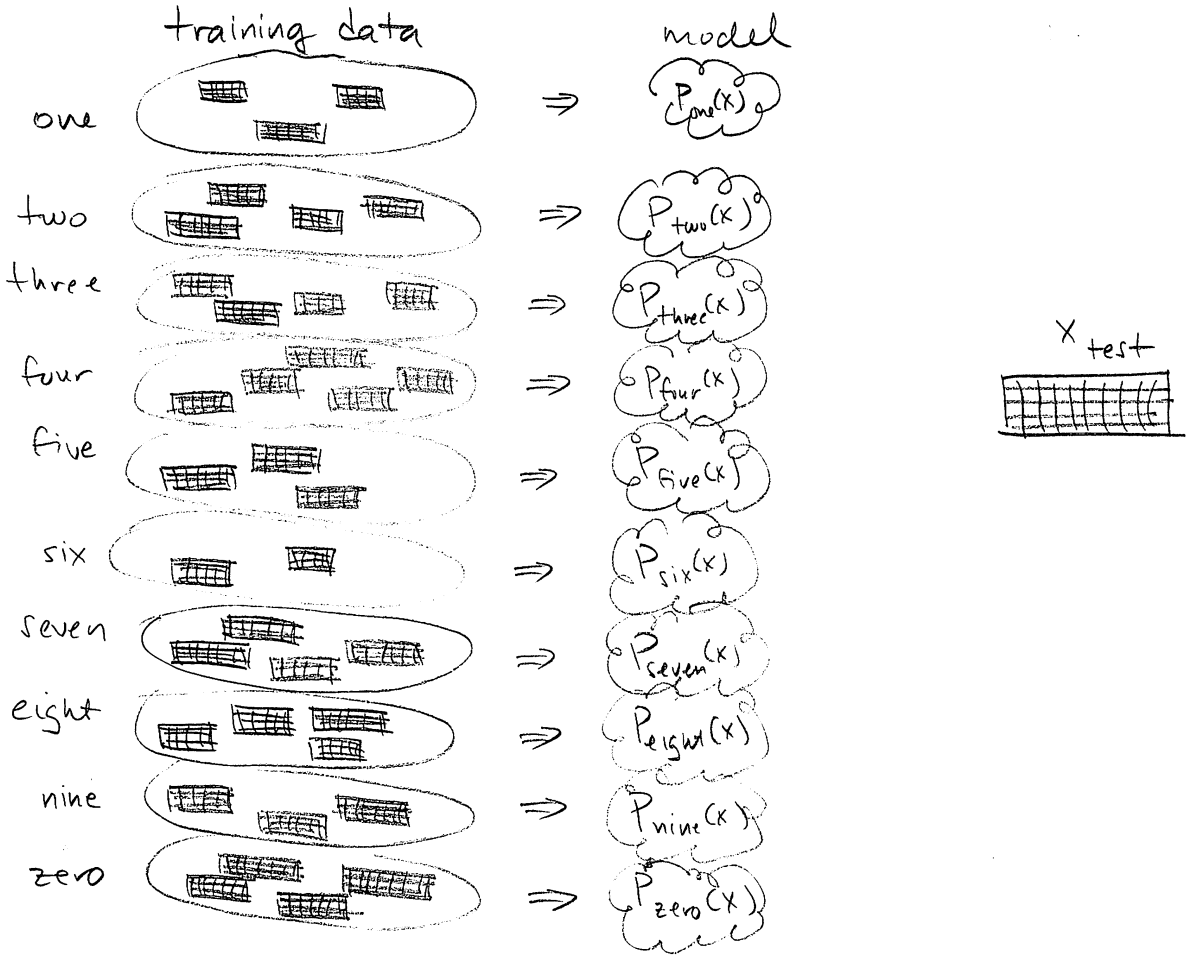
$$p_a = \frac{\text{(count of } a\text{)}}{\sum_{\text{src}(a')=\text{src}(a)} \text{(count of } a'\text{)}} = \frac{\sum_t \gamma(a,t)}{\sum_{\text{src}(a')=\text{src}(a)} \sum_t \gamma(a',t)}$$

- *e.g.*, single Gaussian, mean $\mu_{a,d}$ for dim d.

$$\mu_{a,d} = \text{(mean weighted by } \gamma(a,t)) = \frac{\sum_t \gamma(a,t) x_{t,d}}{\sum_t \gamma(a,t)}$$

Where Are We?

What is Decoding?



$$(\text{answer}) = \arg\max_{\omega \in \text{vocab}} P_\omega(\mathbf{x}_{\text{test}})$$

What Algorithm?

$$(\text{answer}) = \arg\max_{\omega \in \text{vocab}} P_\omega(\mathbf{x}_{\text{test}})$$

- For each word ω, how to compute $P_\omega(\mathbf{x}_{\text{test}})$?
- Forward or Viterbi algorithm.

What Are We Trying To Compute?

$$
\begin{aligned}
P(\mathbf{x}) &= \sum_{\text{paths } A} P(\mathbf{x}, A) \\
&= \sum_{\text{paths } A} \prod_{t=1}^{T} p_{a_t} \times P(\vec{x}_t | a_t) \\
&= \sum_{\text{paths } A} \prod_{t=1}^{T} (\text{arc cost})
\end{aligned}
$$

Dynamic Programming

- Shortest path problem.

$$(\text{answer}) = \min_{\text{paths } A} \sum_{t=1}^{T_A} (\text{edge length})$$

- Forward algorithm.

$$P(\mathbf{x}) = \sum_{\text{paths } A} \prod_{t=1}^{T} (\text{arc cost})$$

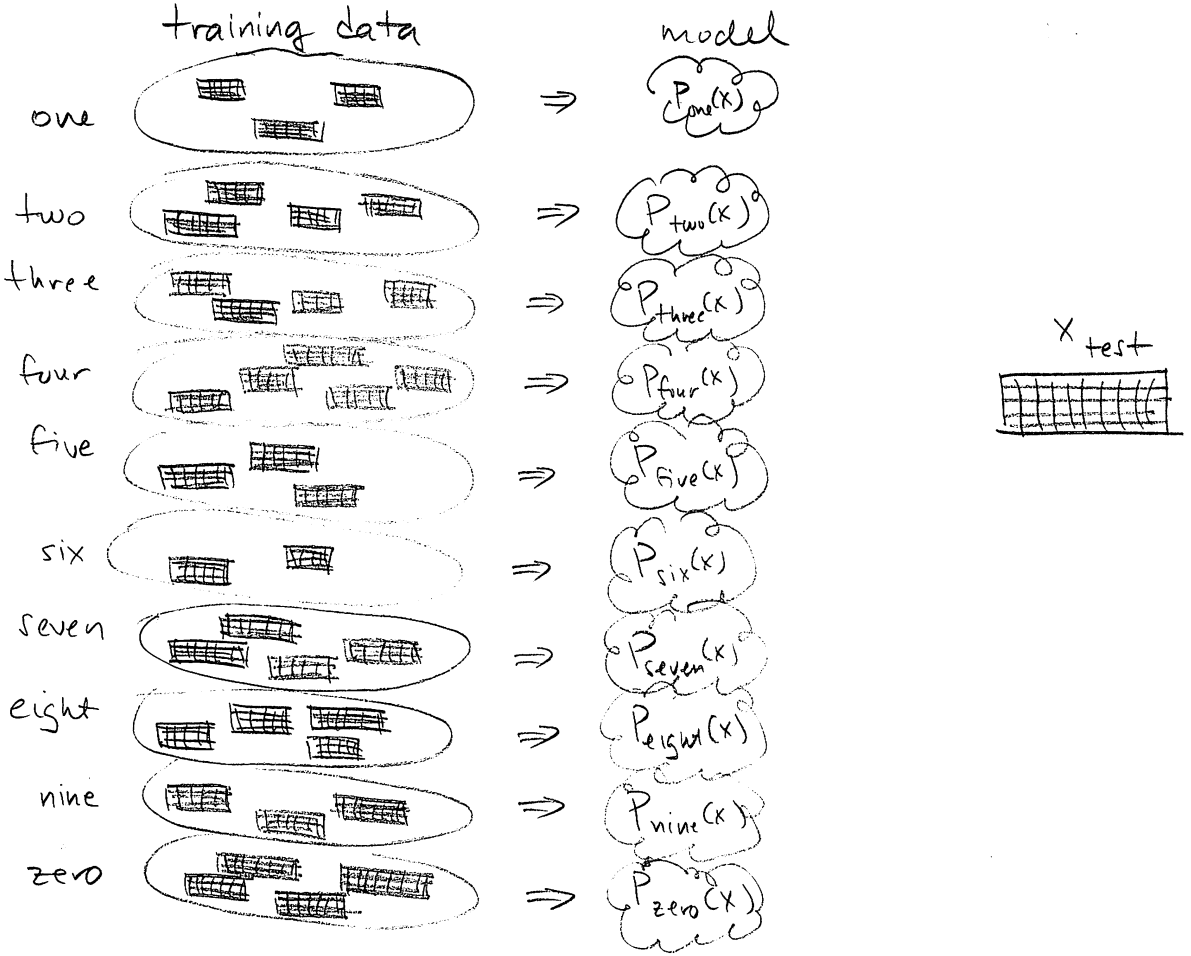
- Viterbi algorithm.

$$P(\mathbf{x}) \approx \max_{\text{paths } A} \prod_{t=1}^{T} (\text{arc cost})$$

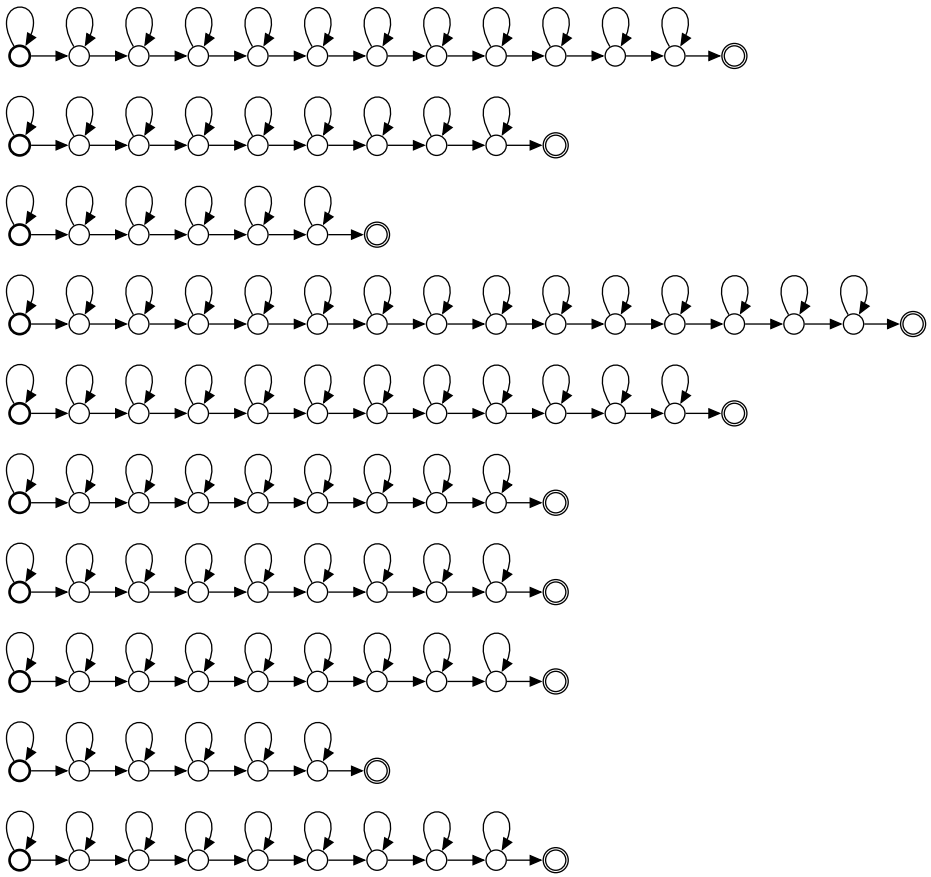
- Any semiring will do.

Scaling

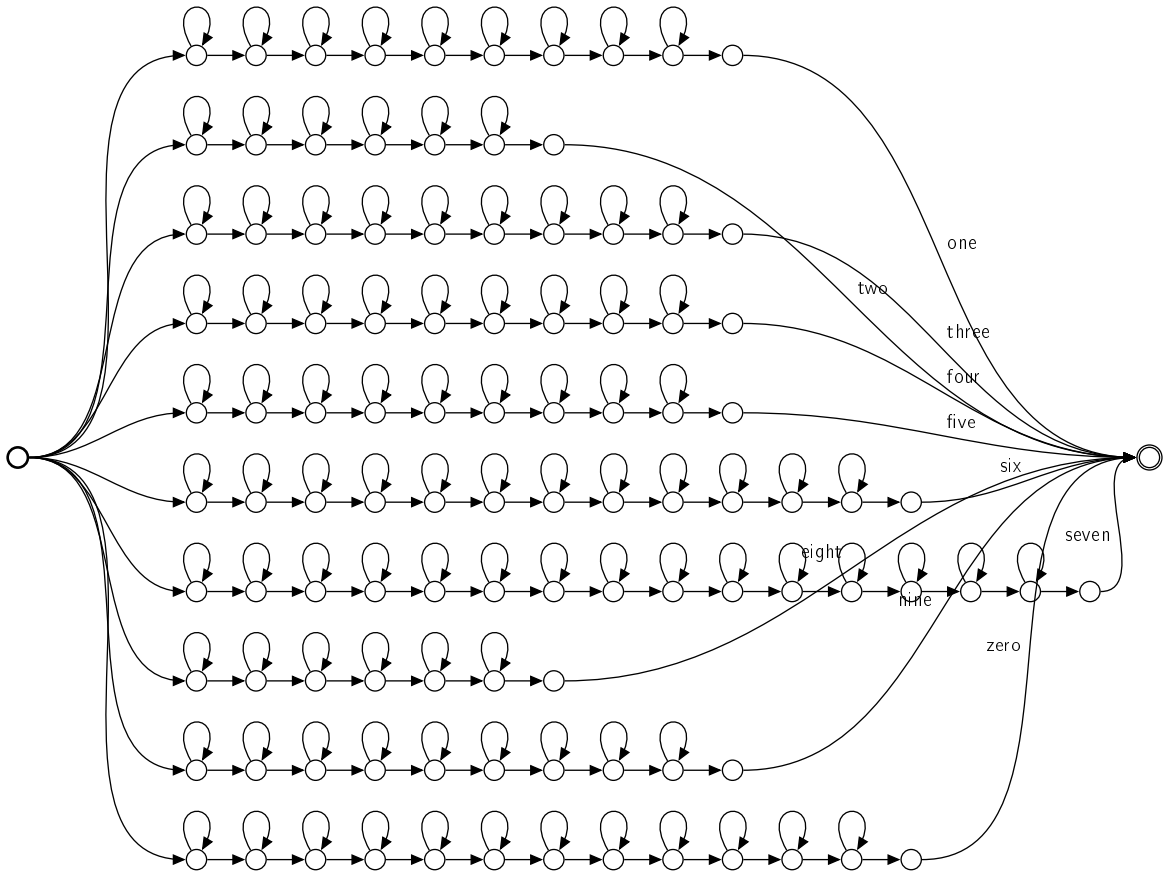
- How does decoding time scale with vocab size?



The One Big HMM Paradigm: Before



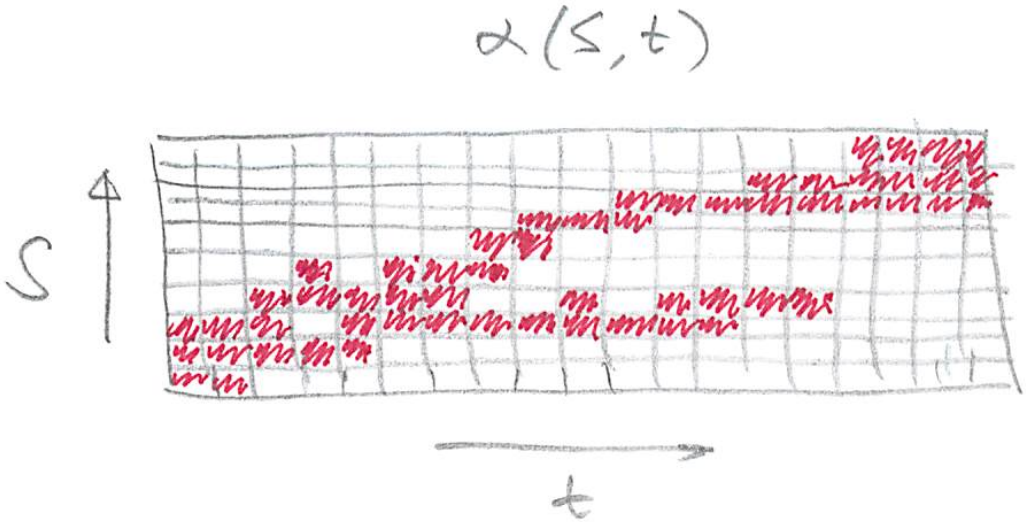
The One Big HMM Paradigm: After



- How does this help us?

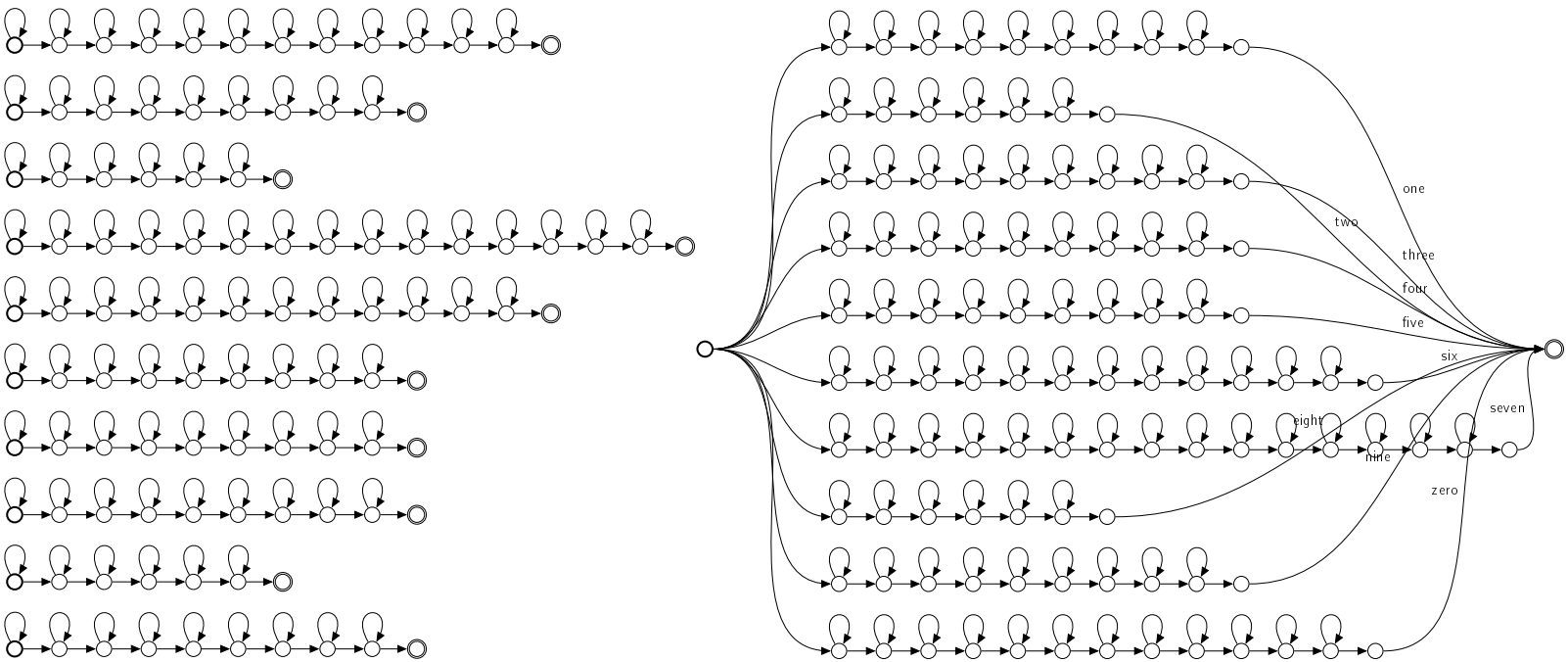
Pruning

- What is time complexity of Forward/Viterbi?
  - How many values $\alpha(S, t)$ to fill?
- Idea: only fill k best cells at each frame.
  - What is time complexity?
  - How does this scale with vocab size?



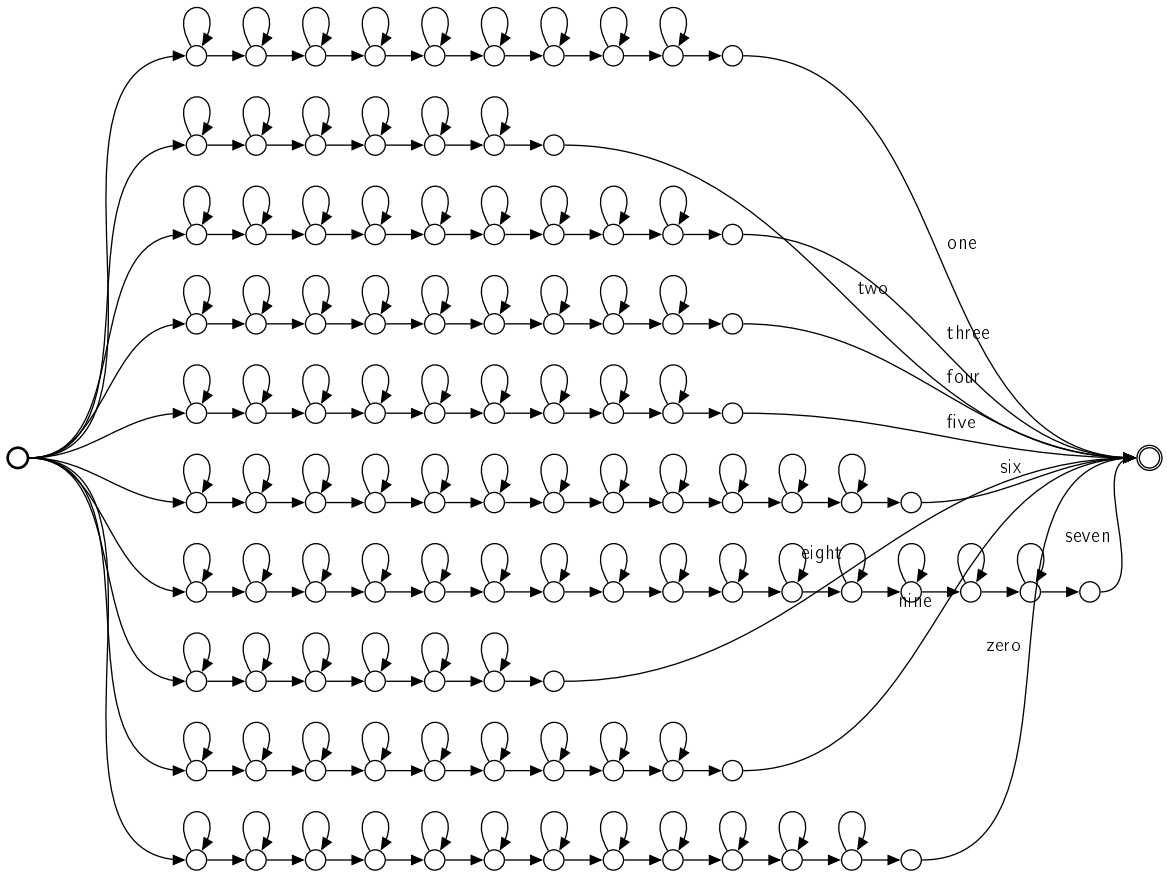
How Does This Change Decoding?

- Run Forward/Viterbi *once*, on one big HMM . . .
  - Instead of once for every word model.
- Same algorithm; different graph!

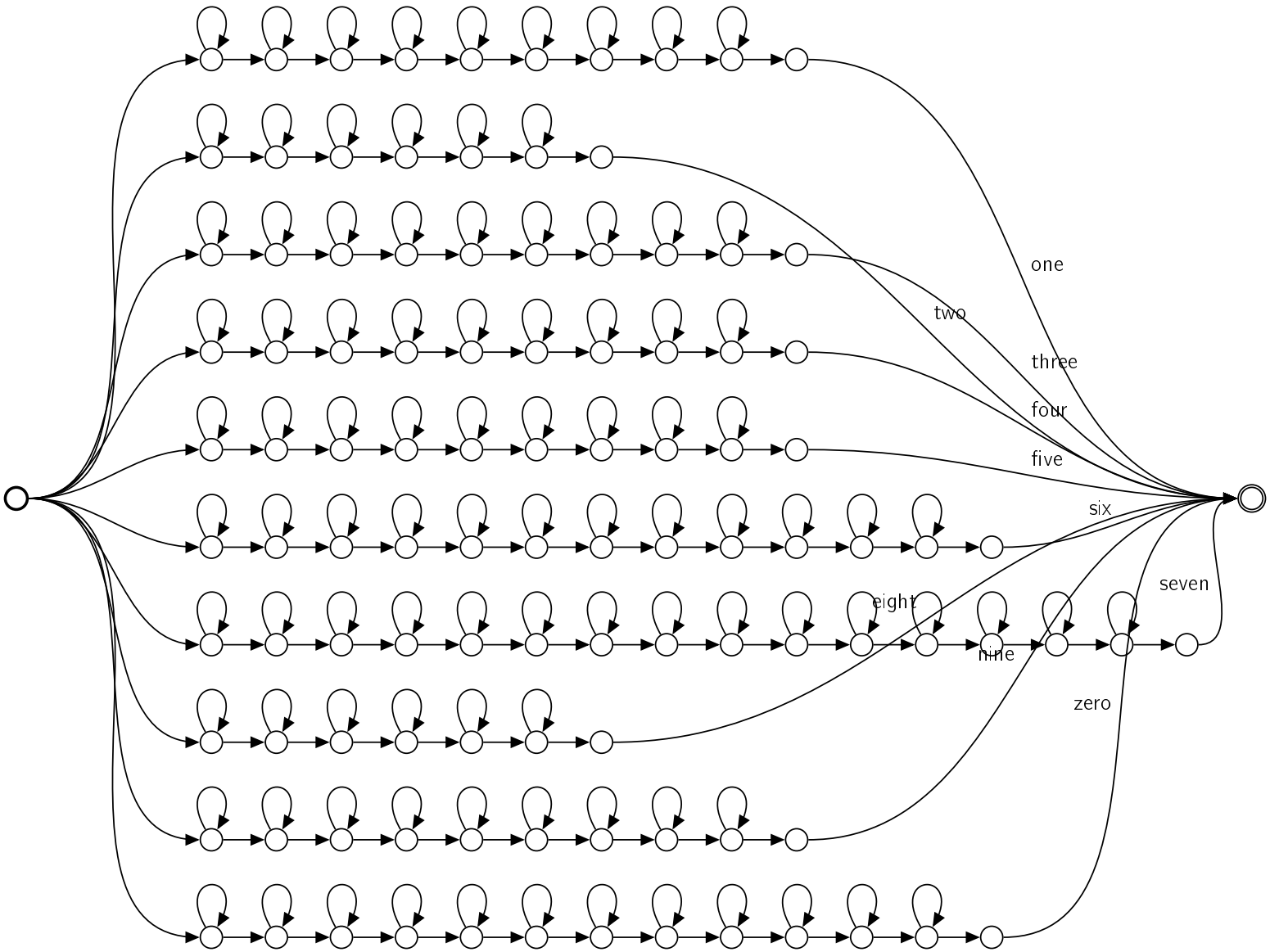


Forward or Viterbi?

- What are we trying to compute?
  - Total prob? Viterbi prob? Best word?



Recovering the Word Identity



Where Are We?

Hyperparameters

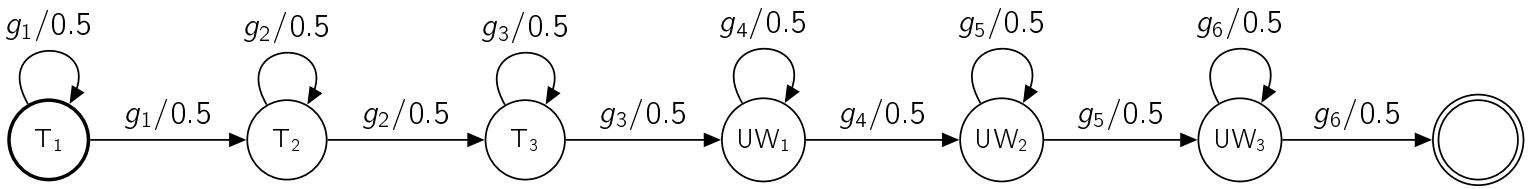
- What is a *hyperparameter*?
  - A tunable knob or something adjustable . . .
  - That can't be estimated with "normal" training.
- Can you name some?
  - Number of states in each word HMM.
  - HMM topology.
  - Number of GMM components.

“Estimating” Hyperparameters

- How does one set hyperparameters?
- Just try different values $\Rightarrow$ expensive!
  - Testing value $\Rightarrow$ train whole HMM/GMM system.
- What criterion to optimize?
  - Normal parameter: Likelihood (smooth).
  - Hyperparameters: Word-error rate (noisy).
  - Gradient descent unreliable; grid search instead.
- Ask an old-timer.
- What are good hyperparameter settings for ASR?

How Many States?

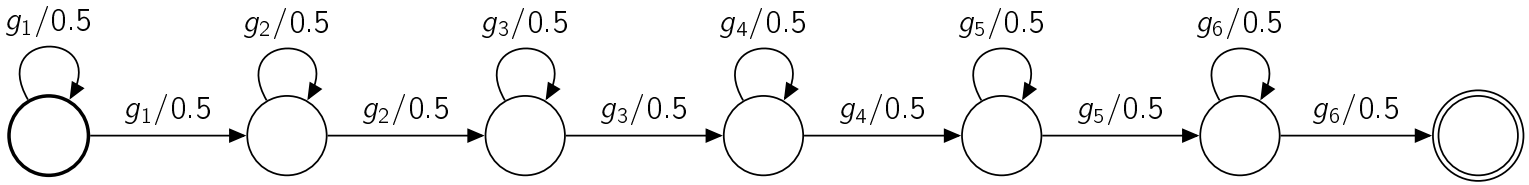
- Rule of thumb: three states per phoneme.
  - A phoneme has a start, middle, and end?
- Example: TWO is composed of phonemes T UW.
  - Two phonemes $\Rightarrow$ six HMM states.



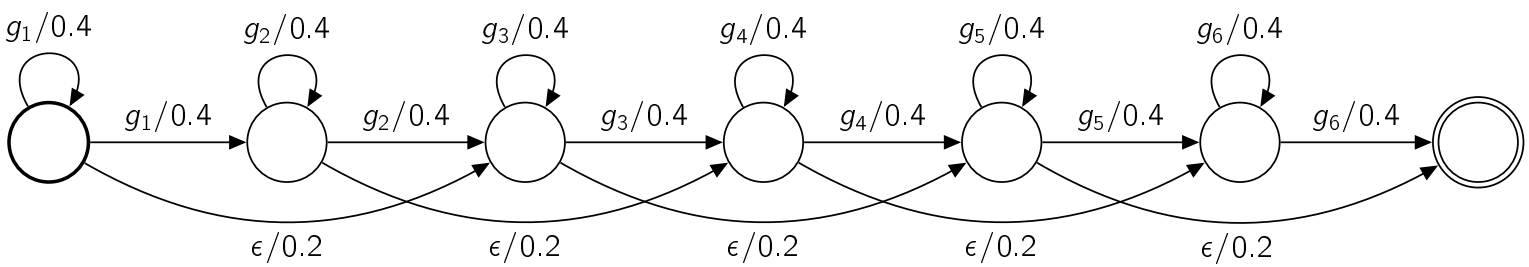
- What guarantee each state models intended sound?

Which HMM Topology?

- A standard topology.
  - Must say sounds of word in order.
  - Can stay at each sound indefinitely.



- Can we skip sounds, *e.g.*, *fifth*?
  - Use *skip arcs* $\Leftrightarrow$ arcs with no output.
  - Need to modify Forward, Viterbi, etc.



The Smallest Number in the World

- Demo.

Probabilities and Log Probabilities

$$P(\mathbf{x}) = \sum_{\text{paths } A} \prod_{t=1}^{T} p_{a_t} \sum_{\text{comp } j} p_{a_t,j} \prod_{\text{dim } d} \mathcal{N}(x_{t,d}; \mu_{a_t,j,d}, \sigma^2_{a_t,j,d})$$

- 1 sec of data $\Rightarrow T = 100 \Rightarrow$ Multiply 4,000 likelihoods.
  - Easy to generate values below 10^{-307}.
  - Cannot store in C/C++ 64-bit double.
- What to do?
- Solution: store log probs instead of probs.
  - Compute $\log \alpha(S, t)$, not $\alpha(S, t)$.

Viterbi Easy, Forward Tricky

- Viterbi algorithm

$$\hat{\alpha}(S,t) = \max_{S' \xrightarrow{x_t} S} P(S' \xrightarrow{x_t} S) \times \hat{\alpha}(S', t-1)$$

$$\log \hat{\alpha}(S,t) = \max_{S' \xrightarrow{x_t} S} \left[\log P(S' \xrightarrow{x_t} S) + \log \hat{\alpha}(S', t-1) \right]$$

- Forward algorithm

$$\alpha(S,t) = \sum_{S' \xrightarrow{x_t} S} P(S' \xrightarrow{x_t} S) \times \alpha(S', t-1)$$

$$\log \alpha(S,t) = \log \sum_{S' \xrightarrow{x_t} S} \exp \left[\log P(S' \xrightarrow{x_t} S) + \log \alpha(S', t-1) \right]$$

- See Holmes, p. 153–154.

What Have We Learned So Far?

- Single-word recognition.
  - How to recognize a *single* word.
  - *e.g.*, can handle a digit, not a digit string.
- What's this good for?
  - Old-time voice dialing.
  - Recognizing digits in old-time phone menus.
  - Not much else.

Part II

Continuous Word ASR

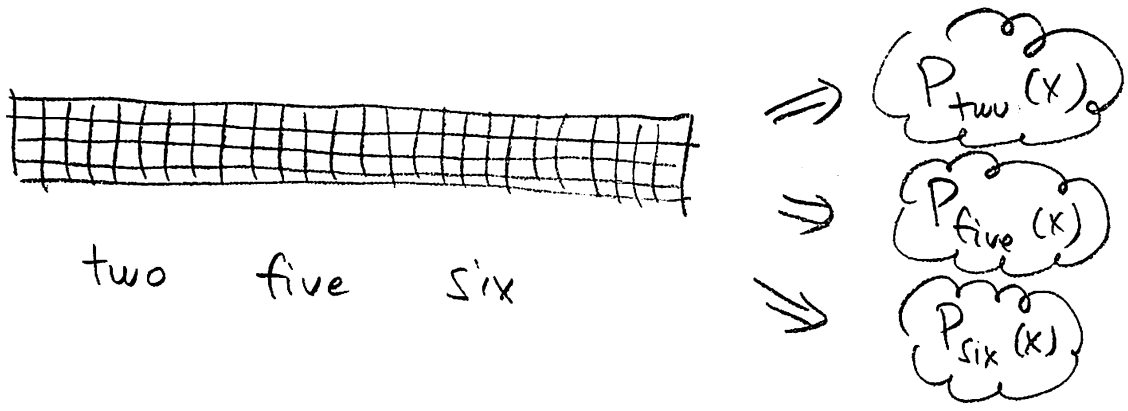
Where Are We?

Next Step: Isolated Word Recognition

- It's . . . when . . . you . . . talk . . . like . . . this.

Training Data, Test Data

- What you need: silence-based segmenter.
- Chop up training data.



- Chop up test data.
- Reduces to single-word ASR.

Continuous Word Recognition

- It's when you talk like this.

Continuous Word Recognition

- Single word.
  - Find word model with highest prob:

$$\arg\max_{\omega \in \text{vocab}} P_\omega(\mathbf{x}_{\text{test}})$$

  - V-way classification.
- Continuous word.
  - Find word *sequence* with highest prob:

$$\arg\max_{\omega \in \text{vocab}^*} P(\mathbf{x}_{\text{test}} | \omega)$$

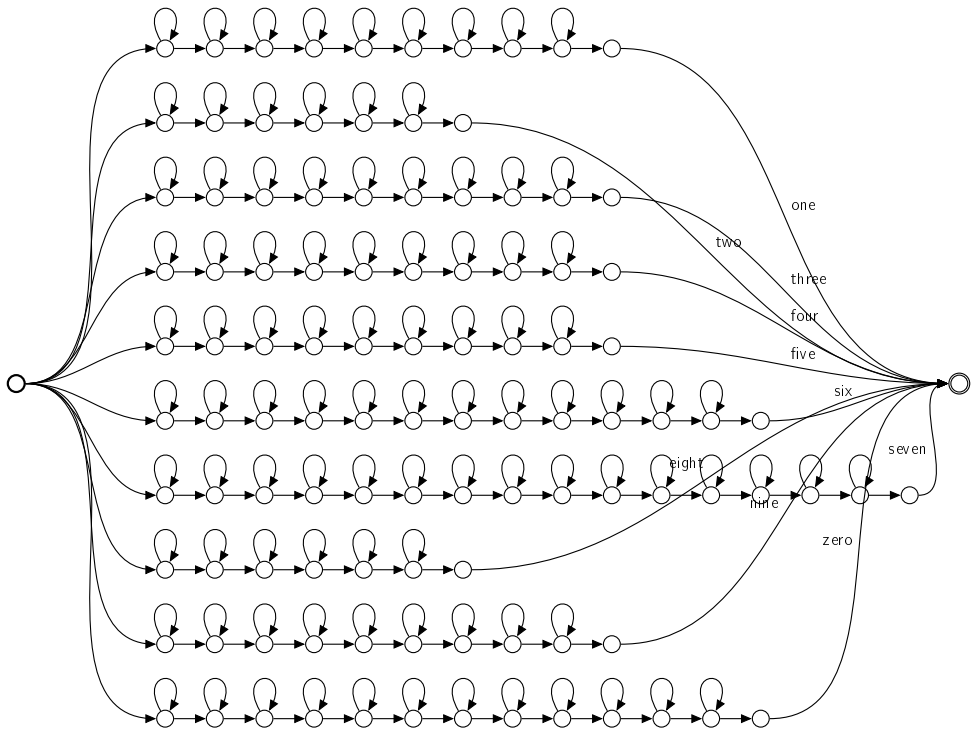
  - ∞-way classification.
- This sounds hard.

Decoding Continuous Word Data

- Have single-word models $P_\omega(\mathbf{x})$.
- How to decode continuous words?

One Big HMM Paradigm

- How to modify HMM . . .
  - To accept word sequences instead of single words?



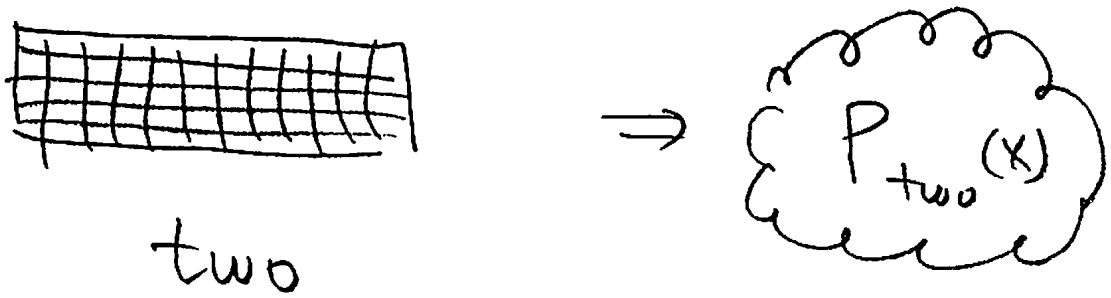
- Loop!

What Do We Need To Change in Viterbi?

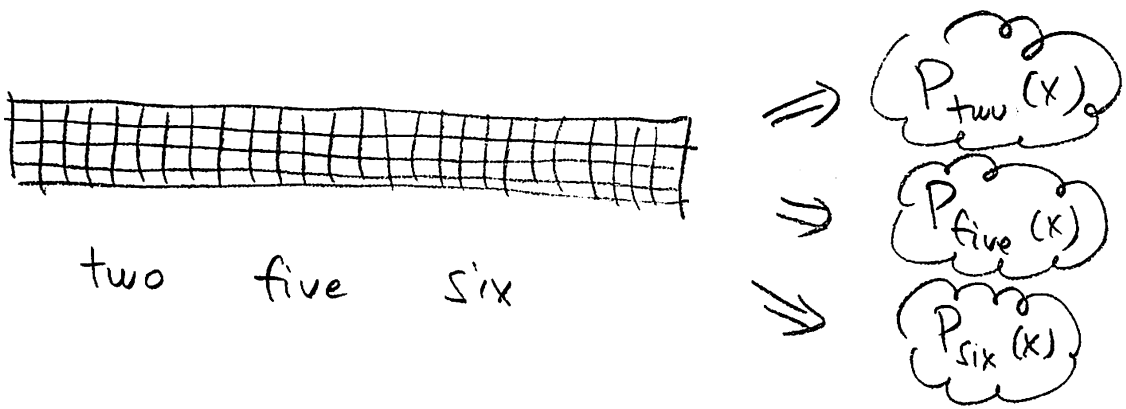
- Nada.

Training on Continuous Word Data

- Isolated word.



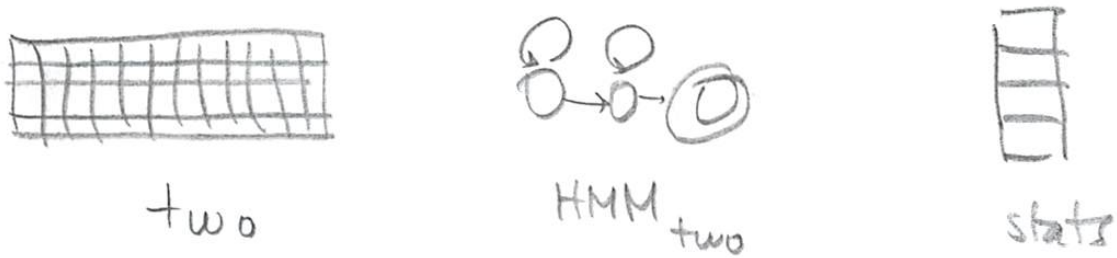
- Continuous word.



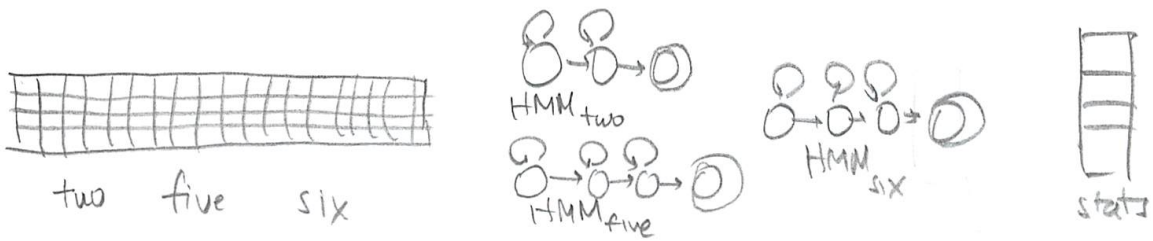
- Don't know where words begin and end!

How Does Training Work Again?

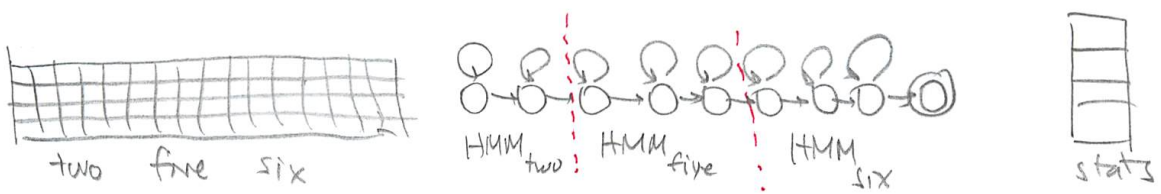
- Isolated word.



- Continuous word: what to do?



- Idea: concatenate HMM's!



What Do We Need To Change in FB?

- Nada.

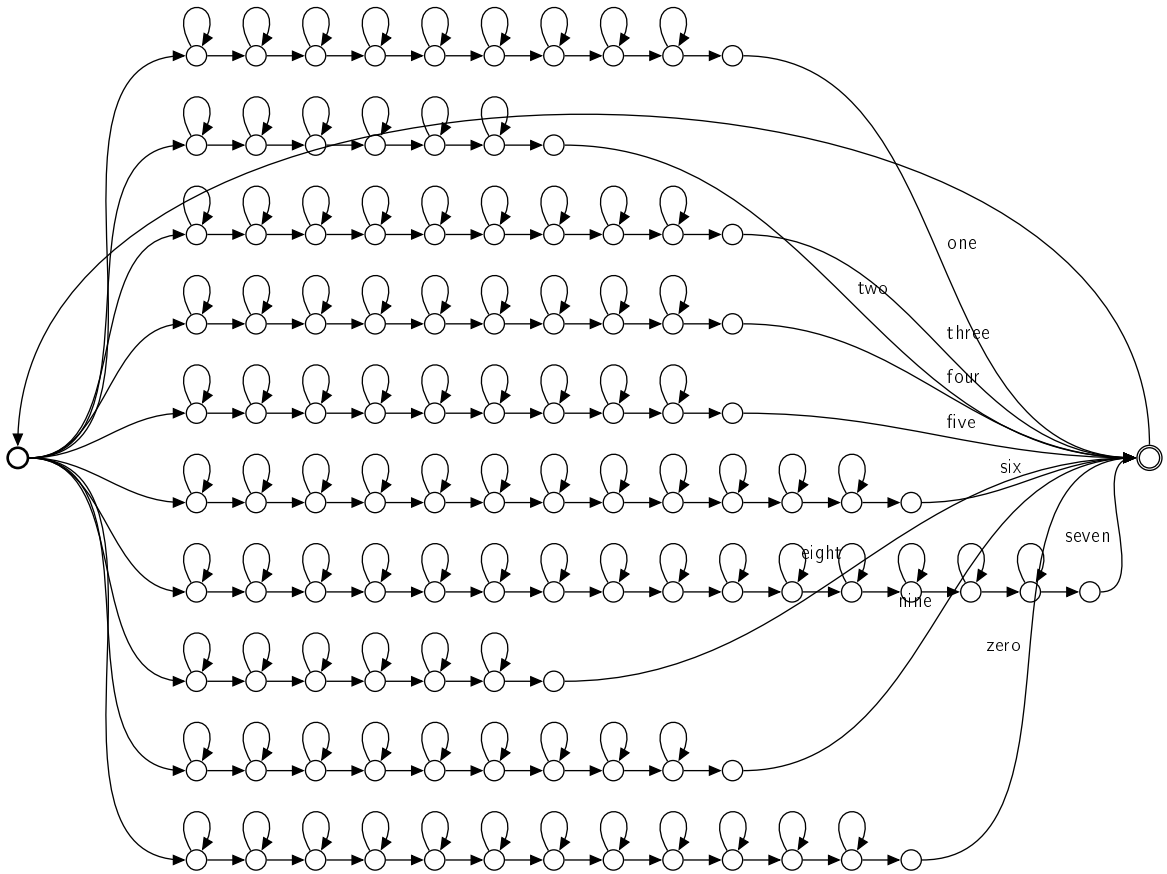
Recap: Continuous Word ASR

- Use “one big HMM” paradigm for decoding.
- Modify HMM’s for decoding and training in intuitive way.
- Everything just works!

Where Are We?

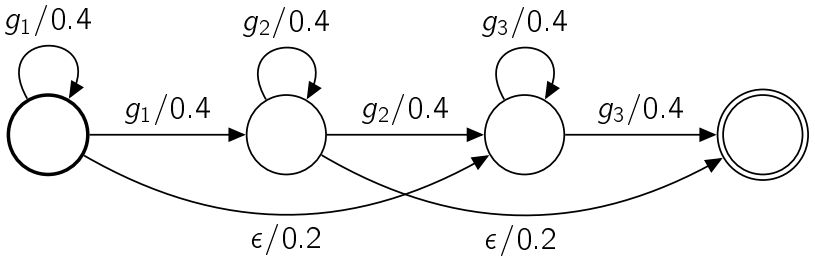
One More Thing

- What happens if we feed isolated speech . . .
  - Into our continuous word system?



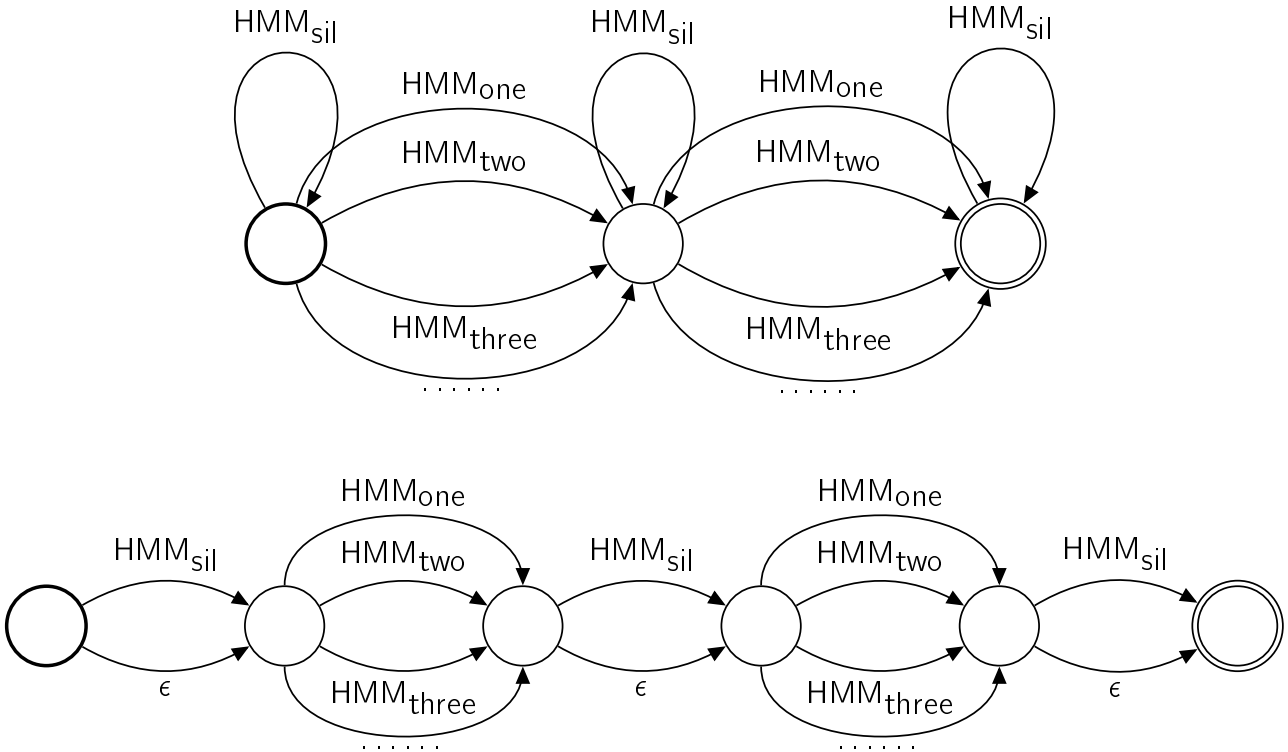
What To Do About Silence?

- Treat silence as just another word ($\sim$SIL).
- How to design HMM for silence?



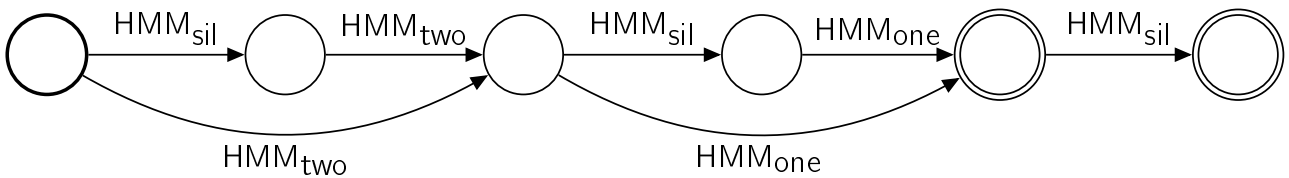
Silence In Decoding

- Where *may* silence occur?
- How many silences can occur in a row?
- Rule of thumb: unnecessary freedom should be avoided.
  - cf. Patriot Act.



Silence In Training

- Usually not included in transcripts.
- *e.g.*, HMM for transcript: ONE TWO



- Lab 2: graphs constructed for you.

Recap: Silence

- Don’t forget about silence!
- Silence can be modeled as just another word.
- Generalization: noises, music, filled pauses.

Where Are We?

HMM/GMM Systems Are Easy To Use

- List of inputs.
- Hyperparameters.
  - HMM topology, # states; # GMM components.
- What else?*
- Utterances with transcripts.
  - Automatically induces word begin/ends, silences.
- Period.

*Small vocabulary only.

HMM/GMM Systems Are Flexible

- Same algorithms for:
  - Single word, isolated word, continuous word ASR.
- Just change how HMM is created!

HMM/GMM Systems Are Scalable

- As training data, vocabulary grows.
- In decoding speed.
  - Pruning $\Rightarrow$ time grows slowly.*
- In model size.
  - Number of parameters grow slowly.*

*When using large-vocab methods described in next few lectures.

HMM/GMM's Are The Bomb

- State of art since invented in 1980's.
  - That's 30+ years!
- Until a couple years ago ...
  - Basically every production system was HMM/GMM.
  - Most probably still are.

Segue: What Have We Learned So Far?

- Small-vocabulary continuous speech recognition.
- What's this good for?
  - Digit strings.
  - Not much else.
- What's next: large-vocabulary CSR.

Part III

Language Modeling

Where Are We?

Demo

What's the Point?

- ASR works better if you say something "expected".
- Otherwise, it doesn't do that well.

Wreck a Nice Peach?

- Demo.

  THIS IS OUR ROOM FOR A FOUR HOUR PERIOD .
  THIS IS HOUR ROOM FOUR A FOR OUR . PERIOD

  IT IS EASY TO RECOGNIZE SPEECH .
  IT IS EASY TO WRECK A NICE PEACH .

- Homophones; acoustically ambiguous speech.
- How does it get it right . . .
  - Even though acoustics for pair is same?
  - (What if want other member of pair?)
- Need to model “expected” word sequences!

How Do We Do Recognition?

- $\mathbf{x}_{\text{test}}$ = test features; $P(\mathbf{x}|\omega)$ = HMM/GMM model.

$$\text{(answer)} = ???$$

$$\text{(answer)} = \arg\max_{\omega \in \text{vocab}^*} P(\mathbf{x}_{\text{test}}|\omega)$$

- Return the word sequence that . . .
- Assigns the highest prob to the utterance.

Does This Prefer Likely Word Sequences?

- *e.g.*, $P(\mathbf{x}_{\text{test}}|\text{OUR ROOM})$ *vs.* $P(\mathbf{x}_{\text{test}}|\text{HOUR ROOM})$.
- If I say `AA R R UW M`, how do these compare?
- They should be about the same.

How Do We Fix This?

- Want term $P(\omega)$.
  - Prior over word sequences; prefers likely sequences.
- What HMM/GMM's give us: $P(\mathbf{x}|\omega)$.
- Old: word sequence that maximizes likelihood of feats.

$$\text{(answer)} = \arg\max_{\omega} \; P(\mathbf{x}|\omega)$$

- Idea: most likely word sequence *given* feats!?

$$\text{(answer)} = \arg\max_{\omega} \; P(\omega|\mathbf{x})$$

Bayes' Rule

- The rule:

$$P(\mathbf{x}, \omega) = P(\omega)P(\mathbf{x}|\omega) = P(\mathbf{x})P(\omega|\mathbf{x})$$

$$P(\omega|\mathbf{x}) = \frac{P(\omega)P(\mathbf{x}|\omega)}{P(\mathbf{x})}$$

- Substituting:

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

The Fundamental Equation of ASR

- Old way:

$$\text{(answer)} = \arg\max_{\omega} \; P(\mathbf{x}|\omega)$$

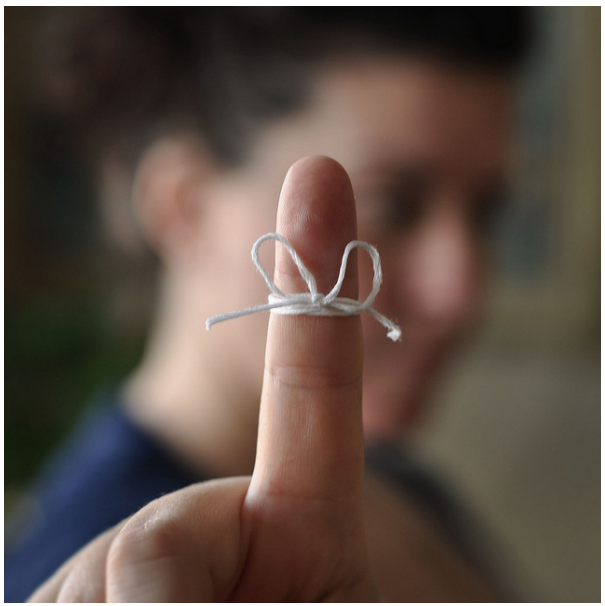
- New way:

$$\text{(answer)} = \arg\max_{\omega} \; P(\omega|\mathbf{x}) = \arg\max_{\omega} \; P(\omega)P(\mathbf{x}|\omega)$$

- Added term $P(\omega)$, just like we wanted.

Remember This!

$$
\begin{aligned}
\text{(answer)} &= \arg\max_{\omega} \ \text{(language model)} \times \text{(acoustic model)} \\
&= \arg\max_{\omega} \ \text{(prior prob over words)} \times P(\text{feats}|\text{words}) \\
&= \arg\max_{\omega} \ P(\omega)P(\mathbf{x}|\omega)
\end{aligned}
$$



Does This Fix Our Problem?

$$\begin{aligned}(\text{answer}) &= \arg\max_{\omega} \ (\text{language model}) \times (\text{acoustic model}) \\ &= \arg\max_{\omega} \ P(\omega)P(\mathbf{x}|\omega)\end{aligned}$$

- What about homophones?

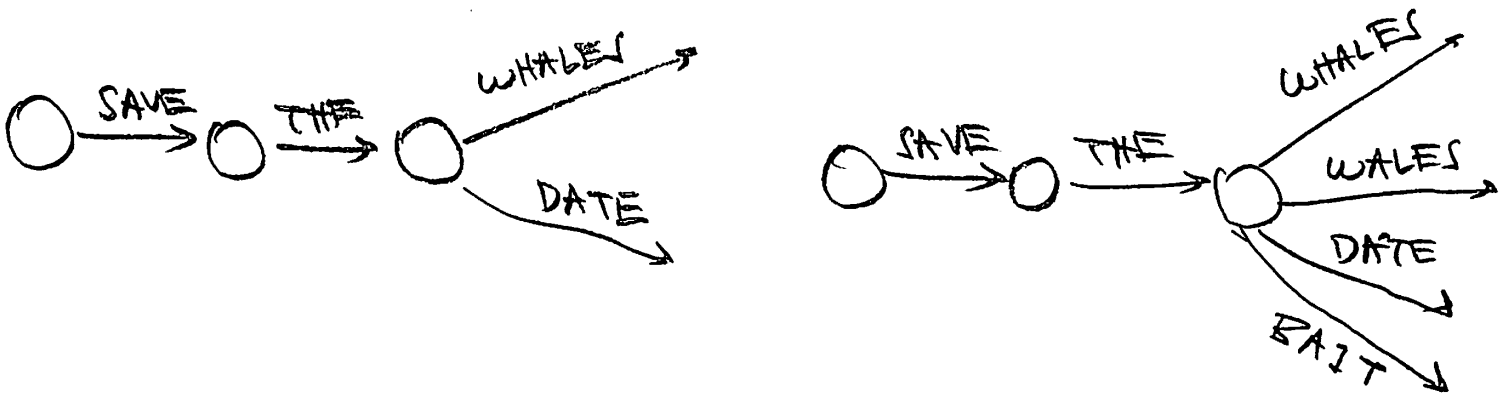
  THIS IS OUR ROOM FOR A FOUR HOUR PERIOD .
  THIS IS HOUR ROOM FOUR A FOR OUR . PERIOD

- What about confusable sequences in general?

  IT IS EASY TO RECOGNIZE SPEECH .
  IT IS EASY TO WRECK A NICE PEACH .

Language Modeling: Goals

- Describe which word sequences are *likely*.
- Eliminate nonsense; restrict choices given to AM.
  - The fewer choices, the better you do!
- Save acoustic model's ass.



Pop Quiz

- What is the fundamental equation of ASR?

Part IV

Epilogue

What's Next

- Language modeling: on the road to LVCSR.
- Lecture 6: Pronunciation modeling.
  - Acoustic modeling for LVCSR.
- Lectures 7, 8: Training, finite-state transducers, search.
  - Efficient training and decoding for LVCSR.

Course Feedback

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