Lecture 5

The Big Picture/Language Modeling

Michael Picheny, Bhuvana Ramabhadran, Stanley F. Chen

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

08 October 2012

Administrivia

- Clear (10); mostly clear (7); unclear (6).
  - Please ask questions!
- Pace: fast (9); OK (6); slow (1).
- Feedback (2+ votes):
  - More/better examples (4).
  - Talk louder/clearer/slower (4).
  - End earlier (2).
  - Too many slides (2).
- Muddiest: Forward-Backward (3); continuous HMM's (2); HMM's in general (2); ...

Administrivia

- 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 (Wednesday, Oct. 17) at 6pm.
  - Start early! Avail yourself of Courseworks.
- Optional non-reading projects.
  - Will post soon; submit proposal in two weeks.

Recap: The Probabilistic Paradigm for ASR

- Notation:
  - $\mathbf{x}$ — observed data, *e.g.*, MFCC feature vectors.
  - ω — word (or word sequence).
- Training: For each word ω, build model $P_\omega(\mathbf{x})$ ...
  - Over sequences of 40d feature vectors $\mathbf{x}$.
- Testing: Pick word that assigns highest likelihood ...
  - To test data $\mathbf{x}_{\text{test}}$.

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

- Which probabilistic model?

Part I

The HMM/GMM Framework

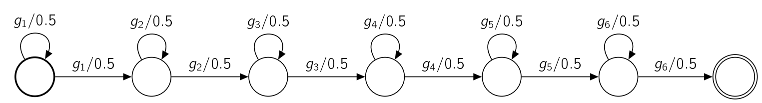
Where Are We?

The Basic Idea

- Use separate HMM to model each word.
- Word is composed of sequence of "sounds".
  - *e.g.*, *BIT* is composed of sounds "*B*", "*IH*", "*T*".
- Use HMM to model which sounds follow each other.
  - *e.g.*, first, expect features for "*B*" sound, . . .
  - Then features for "*IH*" sound, etc.
- For each sound, use GMM's to model likely feature vectors.
  - *e.g.*, what feature vectors are likely for "*B*" sound.

What is an HMM?

- Has states S and arcs/transitions a.
- Has *start* state S_0 (or start distribution).
- Has *transition* probabilities p_a.
- Has *output* probabilities $P(\vec{x}|a)$ on arcs (or states).
  - Discrete: multinomial or single output.
  - Continuous: GMM or other.



What Does an HMM Do?

- Assigns probabilities $P(\mathbf{x})$ to observation sequences:

$$\mathbf{x} = \vec{x}_1, \ldots, \vec{x}_T$$

- Each $\mathbf{x}$ can be output by many *paths* through HMM.
  - Path consists of sequence of arcs $A = a_1, \ldots, a_T$.
- Compute $P(\mathbf{x})$ by summing over path likelihoods.

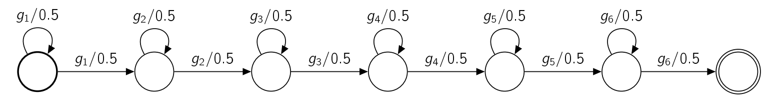
$$P(\mathbf{x}) = \sum_{\text{paths } A} P(\mathbf{x}, A)$$

- Compute path likelihood by ...
  - Multiplying transition and output probs along path.

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

HMM's and ASR

- One HMM per word.
- A standard topology.



- Use diagonal covariance GMM's for output distributions.

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

The Full Model

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

- 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.

The Viterbi and Forward Algorithms

- The Forward algorithm.

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

- The Viterbi algorithm.

$$\text{bestpath}(\mathbf{x}) = \arg\max_{\text{paths } A} P(\mathbf{x}, A)$$

- Can handle exponential number of paths A ...
  - In time linear in number of states, number of frames.*

*Assuming fixed number of arcs per state.

Decoding

- Given trained HMM for each word ω.
- Use Forward algorithm to compute $P_\omega(\mathbf{x}_{\text{test}})$ for each ω.
- Pick word that assigns highest likelihood.

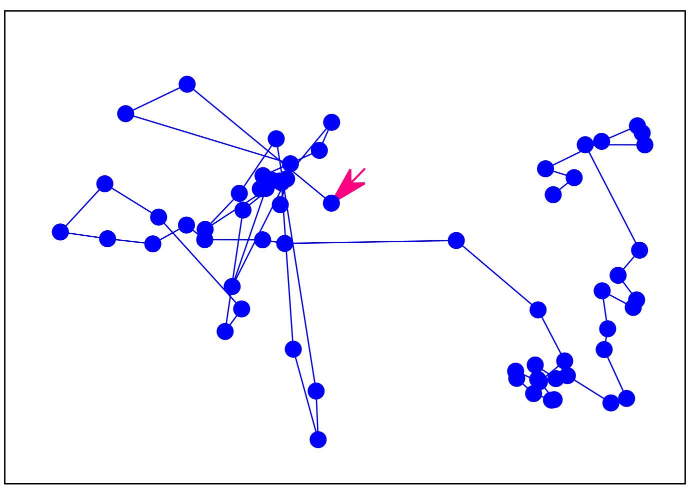
$$\omega^* = \arg\max_{\omega \in \text{vocab}} P_\omega(\mathbf{x}_{\text{test}})$$

The Forward-Backward Algorithm

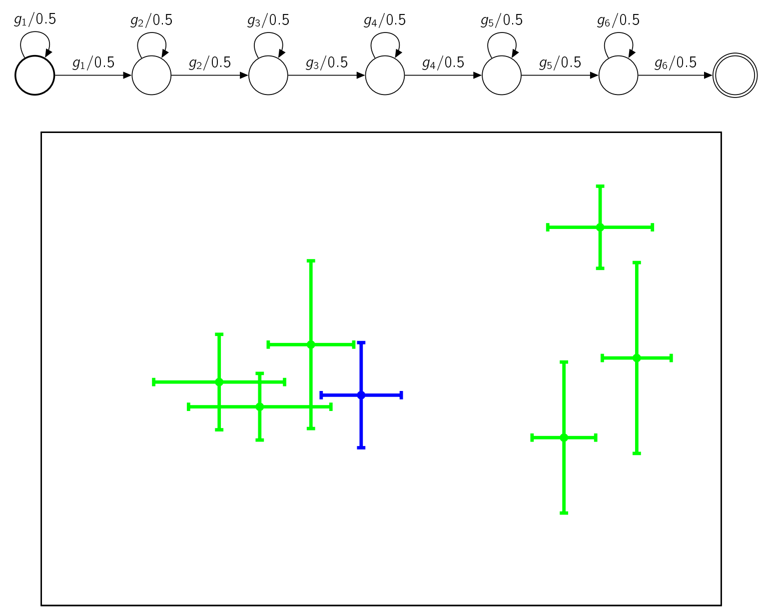
- For each HMM, train parameters $(p_a, p_{a,j}, \mu_{a,j,d}, \sigma^2_{a,j,d})$ ...
  - Using instances of that word in training set.
- Given initial parameter values, ...
  - Iteratively finds local optimum in likelihood.
  - Dynamic programming version of EM algorithm.
- Each iteration linear in number of states, number of frames.
  - May need to do up to tens of iterations.

Example: Speech Data

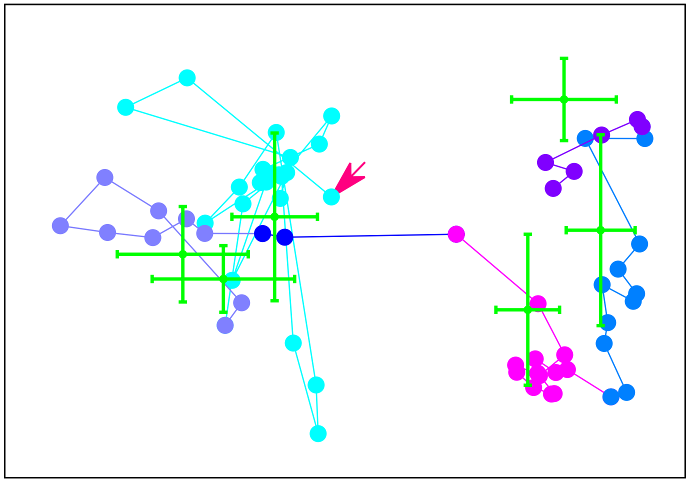
- First two dimensions using Lab 1 front end; the word *TWO*.



Training



The Viterbi Path



Recap

- HMM/GMM framework can model arbitrary distributions . . .
  - Over sequences of continuous vectors.
- Can train and decode efficiently.
  - Forward, Viterbi, Forward-Backward algorithms.

Where Are We?

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.
- Solution: store log probs instead of probs.
  - *e.g.*, in Forward algorithm, instead of storing $\alpha(S, t)$, ...
  - Store values $\log \alpha(S, t)$

Viterbi Algorithm and Max is Easy

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

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

Forward Algorithm and Sum is Tricky

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

$$\begin{aligned}
\log \alpha(S, t) &= \log \sum_{S' \xrightarrow{x_t} S} \exp \left[\log P(S' \xrightarrow{x_t} S) + \log \alpha(S', t-1)\right] \\
&= \log \sum_{S' \xrightarrow{x_t} S} \exp \left[\log P(S' \xrightarrow{x_t} S) + \log \alpha(S', t-1) - C\right] \times e^C \\
&= C + \log \sum_{S' \xrightarrow{x_t} S} \exp \left[\log P(S' \xrightarrow{x_t} S) + \log \alpha(S', t-1) - C\right]
\end{aligned}$$

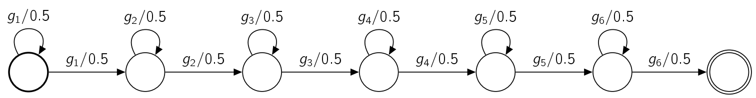
- How to pick C?
- See Holmes, p. 153–154.

Decisions, Decisions ...

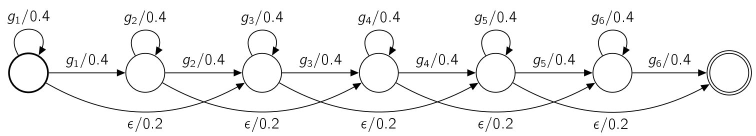
- HMM topology.
- Size of HMM's.
- Size of GMM's.
- Initial parameter values.
- That's it!?

Which HMM Topology?

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

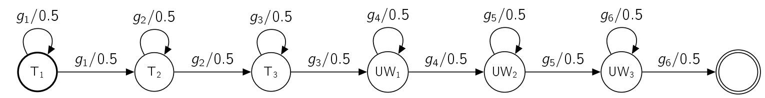


- Can we skip sounds, *e.g.*, *fifth*?
  - Use *skip arcs* $\Leftrightarrow$ arcs with no output.
  - Need to modify Forward, Viterbi, etc.



How Many States?

- Rule of thumb: three states per phoneme.
- Example: TWO is composed of phonemes T UW.
  - Two phonemes $\Rightarrow$ six HMM states.



- No guarantee which sound each state models.
  - States are hidden!

How Many GMM Components?

- Use theory, *e.g.*, Bayesian Information Criterion (lecture 3).
- Just try different values.
  - Maybe 20–40, depending on how much data you have.
- Empirical performance trumps theory any day of week.

Initial Parameter Values: Flat Start

- Transition probabilites p_a — uniform.
- Mixture weights $p_{a,j}$ — uniform.
- Means $\mu_{a,j,d}$ — 0.
- Variances $\sigma^2_{a,j,d}$ — 1.
- Start with single component GMM.
  - Run FB; split each Gaussian every few iters ...
  - Until reach target number of components per GMM.
- This actually works! (More on this in future lecture.)

Recap

- Simple decisions with flat start works!
- Can tune hyperparameters to optimize performance.
  - *e.g.*, skip arcs, number of GMM compnents.
  - Redo this every so often for new domains, forever.
- What happens if too many parameters?
- What happens if too few parameters?

Where Are We?

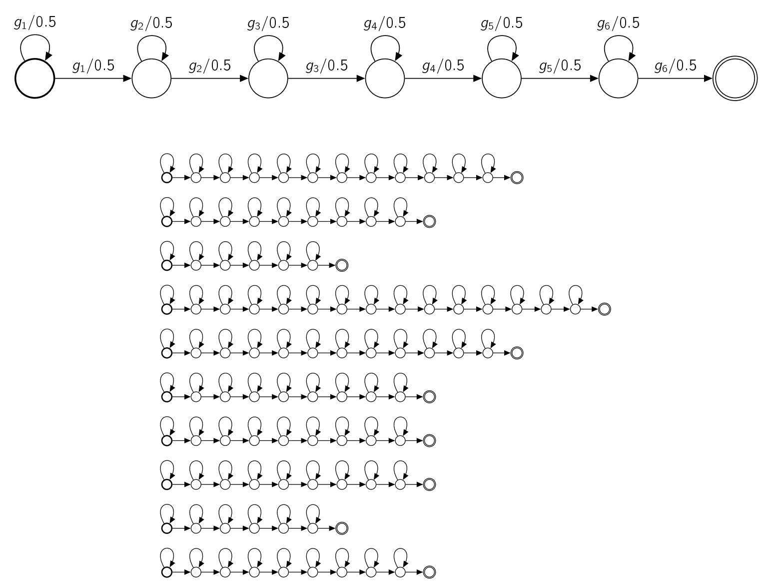
Decoding Secrets Revealed

- What we said:
  - Use Forward algorithm to compute $P_\omega(\mathbf{x}_{\text{test}})$ ...
  - Separately for each word HMM.
  - Pick word that assigns highest likelihood.

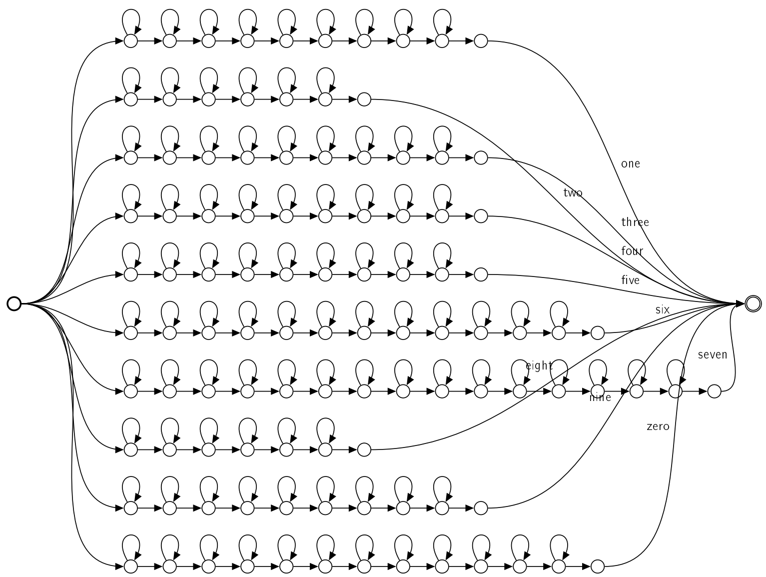
$$\omega^* = \operatorname*{arg\,max}_{\omega \in \text{vocab}} P_\omega(\mathbf{x}_{\text{test}})$$

- Reality:
  - Merge HMM for all words into "one big HMM".
  - Use Viterbi algorithm to find best path given $\mathbf{x}_{\text{test}}$.
  - In backtrace, collect word label on path.

The One Big HMM Paradigm: Before

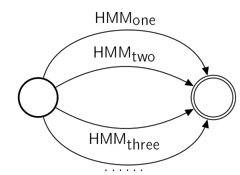


The One Big HMM Paradigm: After



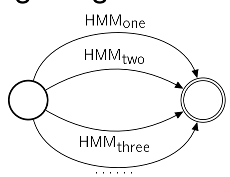
What Have We Gained?

- Pruning (future lecture).
  - *e.g.*, Viterbi algorithm: don't compute every $\hat{\alpha}(S, t)$.
- Graph optimization (future lecture).
  - Can share common prefixes, suffixes between words.
- Easy to extend to continuous word recognition.



From Isolated To Continuous ASR

- Train HMM for each word using isolated word data.
- HMM for decoding: single digit utterance.

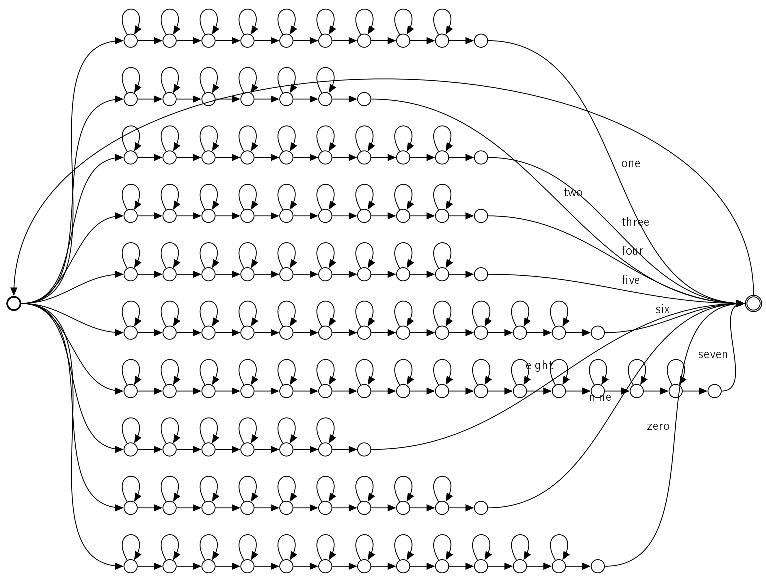


- What HMM to use for two-digit utterances? Three-digit?
- What HMM to allow digit sequences of any length?

From Isolated To Continuous ASR

- Just change topology of decoding HMM ...
  - To reflect word sequences to allow.
- Use Viterbi to find best path as before.
- Attach word labels to each word HMM in big graph.
  - In backtrace, collect word labels along best path.

Recovering the Word Sequence



What About Training?

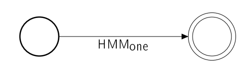
- Old scenario: training data composed of . . .
  - Single digit utterances labeled with single digits.
- New scenario: training data composed of . . .
  - Multiple digit utterances labeled with digit sequences.
- Much easier to collect lots of data.
- Data reflects coarticulation between consecutive words.
- Not told where each digit begins and ends!?

What About Training?

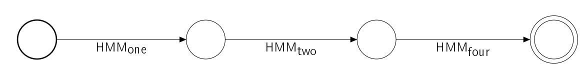
- Old scheme (one iteration of FB):
  - For each utterance, take HMM associated with word.
  - Compute FB counts for parameters in that HMM.
  - Sum counts over data; reestimate parameters.
- New scheme:
  - Construct HMM for utterance in logical way!?

What About Training?

- If transcript is *ONE*, use HMM:



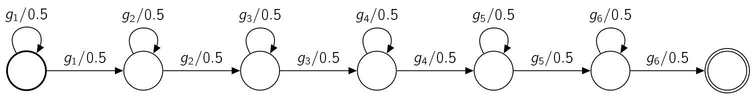
- If transcript is *ONE TWO FOUR*, use HMM:



- Old view: ten HMM's; disjoint parameters.
- New view: lots of HMM's.
  - Shared sub-HMM's and parameters between HMM's.

Parameter Tying

- When same parameter (*e.g.*, p_a, $p_{a,j}$, $\mu_{a,j,d}$, $\sigma^2_{a,j,d}$) ...
  - Used in multiple places.
  - In same HMM, or different HMM's.



- Called *parameter tying*.
  - View: different parameter in each location ...
  - But *tied* to have same value.
- Does EM/Forward-Backward still work?

Parameter Tying and Forward-Backward

- E-step: Compute arc posteriors in same way.
- M-step: ML estimation of parameters given arc posteriors.
  - Log likehood is function only of counts!
  - Doesn't matter if counts collected across ...
  - Different utterances and/or different HMM locations!
  - ML estimate: count and normalize!

Recap: Continuous Word ASR

- Use "one big HMM" paradigm for decoding.
- Modify HMM's for decoding and training in intuitive way.
- Everything just works!
  - All algorithms same; just modify backtrace a little.
  - Forward-Backward still finds good optimum!

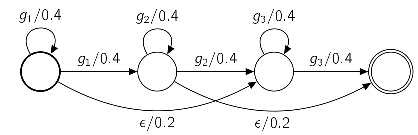
What's Missing?

- Audio sample 1: *2-4-6-3-1*
- Audio sample 2: *2-4-6-3-1*
- What's the difference?

What To Do About Silence?

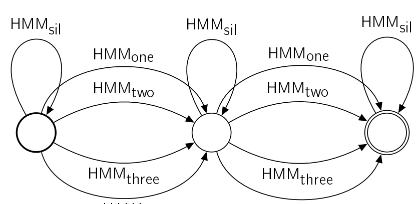
Modeling Silence

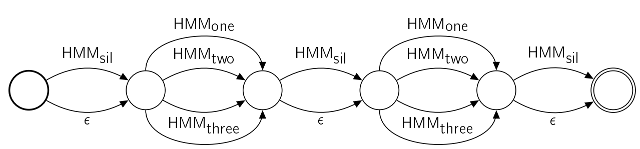
- Treat silence as just another word ($\sim$SIL).
- Not just for modeling silence?
  - Background noise; anything that isn't speech.
- How to design HMM for silence?



Silence In Decoding

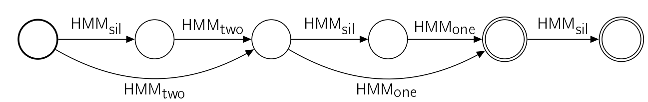
- Where *may* silence occur?
- How many silences can occur in a row?
- Rule of thumb: unnecessary freedom should be avoided.
  - cf. Patriot Act.



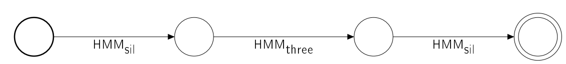


Silence In Training

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



- Silence also used in isolated word training/decoding.
  - Is this necessary?



- Lab 2: graphs constructed for you.

Recap: Silence

- Don't forget about silence!
  - Everyone does sometimes.
- Silence can be modelled as just another word . . .
  - That can occur anywhere.
- Generalization: noises, music, filled pauses.

Where Are We?

Ingredients for HMM/GMM CSR System

- Data.
  - Utterances with transcripts.
- Decisions.
  - For each word, HMM topology and size.
  - Number of components in GMM's.
  - Initial parameter values.
- Period.

Hogwarts Has Course on HMM/GMM's . . .

- Because they are magical!
- Isolated $\Rightarrow$ continuous recognition: the same!
- Forward-Backward can automatically induce . . .
  - Where each word begins and ends in training data.
  - Where silence occurs.
  - How to divide each word into "sounds".
- How crazy is that?
- State of art since invented in 1980's.
  - Almost every current production system is HMM/GMM.

DTW and HMM/GMM's

- Lots of similar ideas.
- Can design HMM such that:*

$$\text{distance}^{\text{DTW}}(\mathbf{x}_{\text{test}}, \mathbf{x}_\omega) \approx -\log P_\omega^{\text{HMM}}(\mathbf{x}_{\text{test}})$$

DTW	HMM
template	HMM
frame in template	state in HMM
DTW alignment	HMM path
local path cost	transition (log)prob
frame distance	output (log)prob
DTW search	Viterbi algorithm

*See Holmes, Sec. 9.13, p. 155.

What Have We Gained? Principles!

- Principles make lots of decisions for you!
  - Fewer ways to screw up!
- What decisions no longer have to make?
  - All parameter values!
  - Local path costs (transition probs).
  - Frame distances (per word, per dimension weighting).
- More data $\Rightarrow$ better performance!!!
  - Maximum likelihood estimates improve!

What Have We Gained? Scalability!

- Easy, principled way to handle continuous ASR.
- Smaller "models".
  - DTW: Store every frame, every instance of every word.
  - HMM: Store GMM parameters for $\sim$15 states/word.
- Faster computation.
  - Proportional to number of states/template frames.
  - Share states between words (*e.g.*, phonetic modeling).
  - Reduces number of states further.
- Scales well to lots of training data; large vocabularies.

What Have We Gained? Generalization!

- DTW: Test sample $\mathbf{x}$ receives high score with word ω ...
  - If $\mathbf{x}$ close to single training instance of ω.
- HMM/GMM: $\mathbf{x}$ receives high score with word ω ...
  - If each sound in $\mathbf{x}$ matches ...
  - Corresponding state in word HMM well.
- *i.e.*, can match well if each *sound* in $\mathbf{x}$ matches ...
  - *Any* instance of ω in training set.

If HMM/GMM's Are So Great . . .

- While HMM/GMM's are state of art . . .
  - ASR performance is far from perfect.
- What's the problem?

The Markov Assumption

- In path, output prob conditioned only on current arc.

$$P(\mathbf{x}, A) = \prod_{t=1}^{T} p_a \times P(\vec{x}_t | a)$$

- Everything need to know about past . . .
  - Encoded in identity of state.
  - *i.e.*, conditional independence of future and past.
- What information *do* we encode in state?
- What information *don't* we encode in state?
  - *i.e.*, what independence assumptions have we made?

Keeping Richer State Information

- Solutions.
  - Increase number of states (exponentially)?
  - Higher-order Markov models?
  - Condition on more stuff; *e.g.*, *graphical models*?
- More states $\Rightarrow$ more parameters.
  - Sparse data leads to poor parameter estimates.
- EM training: finds closest local optimum to starting point.
  - Why does this work for HMM/GMM?
  - How to get hidden states to model what you want?
- Bottom line: No competitor to HMM in sight.

What About GMM's?

- Don't seem like God's gift to probability distributions?
  - Nothing wrong, but not awesome either?
- They've been around for so long.
  - A ton of machinery has been developed for them.
  - *e.g.*, adaptation, discriminative training, . . .
- Recent developments: *deep neural networks*.
  - Still use GMM's for bootstrapping.
- GMM's aren't going to disappear soon.

Part II

Language Modeling

Wreck a Nice Beach?

- 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 BEACH .

- How does it get it right . . .
  - Even though acoustics for pair is same?
  - (What if want other member of pair?)

Maximum Likelihood Classification

- Pick word sequence ω which assigns highest likelihood . . .
  - To test sample $\mathbf{x}$.

$$\omega^* = \arg\max_{\omega} P_{\omega}(\mathbf{x}) = \arg\max_{\omega} P(\mathbf{x}|\omega)$$

- What about $\omega_1 = \text{SAMPLE}$, $\omega_2 = \text{SAM PULL}$?
  - $P(\mathbf{x}|\omega_1) \approx P(\mathbf{x}|\omega_2)$
  - Intuitively, much prefer ω_1 to ω_2.
- Something's missing.

What Do We Really Want?

- What HMM/GMM's give us: $P(\mathbf{x}|\omega)$.

$$\omega^* \stackrel{?}{=} \arg\max_{\omega} \; P(\mathbf{x}|\omega)$$

$$\omega^* \stackrel{!}{=} \arg\max_{\omega} \; P(\omega|\mathbf{x})$$

A Little Math

- Bayes' 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:

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

The Fundamental Equation of ASR

- Old way: maximum likelihood classification.

$$\omega^* = \arg\max_{\omega} P(\mathbf{x}|\omega)$$

- New way: maximum *a posteriori* classification.

$$\omega^* = \arg\max_{\omega} P(\omega|\mathbf{x}) = \arg\max_{\omega} P(\omega)P(\mathbf{x}|\omega)$$

- What's new?
  - Prior distribution $P(\omega)$ over word sequences.
  - How frequent each word sequence ω is.

Does This Fix Our Problem?

$$\omega^* = \arg\max_{\omega} P(\omega)P(\mathbf{x}|\omega)$$

- 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 BEACH .

Terminology

$$\omega^* = \arg\max_{\omega} P(\omega)P(\mathbf{x}|\omega)$$

- $P(\mathbf{x}|\omega)$ = *acoustic model*.
  - Models frequency of acoustic feature vectors $\mathbf{x}$ ...
  - Given word sequence ω.
  - *i.e.*, HMM/GMM's.
- $P(\omega)$ = *language model*.
  - Models frequency of each word sequence ω.
  - The rest of this lecture.

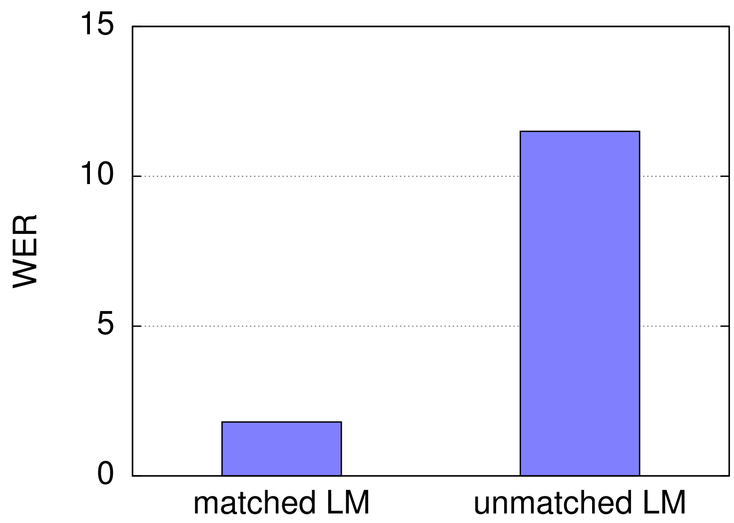
Language Modeling: Goals

- Specific to *domain*!!!
- Describe which word sequences are *allowed*.
  - *e.g.*, restricted domains like digit strings.
- Describe which word sequences are *likely*.
  - *e.g.*, unrestricted domains like web search.
  - *e.g.*, BRITNEY SPEARS *vs.* BRIT KNEE SPEARS.
- Analogy: multiple-choice test.
  - LM restricts choices given to acoustic model.
  - The fewer choices, the better you do.

Real World Toy Example (Untuned)

- Test data: single digits.
- Language model 1: matched.
  - Digit sequences of length 1 equiprobable (10 choices).
- Language model 2: unmatched.
  - Sequences of any length equiprobable (∞ choices).

Real World Toy Example (Untuned)



What Type of Model?

- Want probability distribution over sequence of symbols ...
  - From finite vocabulary.

$$P(\omega) = P(w_1 w_2 \cdots)$$

- Is there some type of model we know can do this?
- Hmm ...

Discrete (Hidden) Markov Models

- What is language model training data?
  - Must match domain!
- *Grammars* — hidden Markov models.
  - Restricted domain.
  - Little or no training data available.
  - *e.g.*, airline reservation app.
- *n-gram models* — Markov models of order $n-1$.
  - Unrestricted domain.
  - Lots of training data available.
  - *e.g.*, web search app.

Grammars for Constrained Domains

- If no LM data available; expensive to create/collect.
  - *e.g.*, name dialer; yellow pages; navigation; moviefone.
- Hack up HMM and parameters as best you can.
  - Using manual or semi-automated methods.
  - Better than using general unconstrained LM.
- Painful, non-robust, non-scalable.
- Automatically learn HMM topology, parameters?
  - Can do some parameter training if enough data?
  - Inducing topology of HMM is open problem.

Where Are We?

Introduction

- Imagine have lots of domain training data.
  - This is true for many domains; *e.g.*, the Web.
- Goal: how to construct Markov model (hidden or not) ...
  - That can take advantage of all this data?
  - And gets better the more data you have?

Idea: Hidden Markov Models

- Like in acoustic modeling.
- What topology?
  - Is there logical topology like for word HMM?
- Learn topology from data?
  - *e.g.*, fully interconnected topology; learn parameters?
- Issues:
  - Local minima issue, FB algorithm.
  - Quadratic in number of states; *e.g.*, 1M states?
- Bottom line: hasn't worked.

Idea: (Non-Hidden) Markov Models

- Review: Markov property order $n-1$ holds if

$$\begin{aligned} P(w_1, \ldots, w_L) &= \prod_{i=1}^{L} P(w_i | w_1, \ldots, w_{i-1}) \\ &= \prod_{i=1}^{L} P(w_i | w_{i-n+1}, \cdots, w_{i-1}) \end{aligned}$$

- *i.e.*, if data satisfies this property ...
  - No loss from just remembering past $n-1$ items!

Markov Model, Order 1: Bigram Model

$$P(w_1, \ldots, w_L) = \prod_{i=1}^{L} P(w_i | w_{i-1}) = \prod_{i=1}^{L} p_{w_{i-1}, w_i}$$

- Separate multinomial $P(w_i | w_{i-1})$ ...
- For each word *history* w_{i-1}.
- Model $P(w_i | w_{i-1})$ with parameter p_{w_{i-1}, w_i}.

Markov Model, Order 2: Trigram Model

$$P(w_1, \ldots, w_L) = \prod_{i=1}^{L} P(w_i | w_{i-2} w_{i-1}) = \prod_{i=1}^{L} p_{w_{i-2}, w_{i-1}, w_i}$$

- Separate multinomial $P(w_i | w_{i-2} w_{i-1})$ ...
- For each bigram *history* $w_{i-2} w_{i-1}$.
- Model $P(w_i | w_{i-2} w_{i-1})$ with parameter $p_{w_{i-2}, w_{i-1}, w_i}$.

Detail: Sentence Begins

$$P(\omega = w_1 \cdots w_L) = \prod_{i=1}^{L} P(w_i | w_{i-2} w_{i-1})$$

- Pad with beginning-of-sentence token: $w_{-1} = w_0 = \triangleright$.

Detail: Sentence Ends

$$P(\omega = w_1 \cdots w_L) = \prod_{i=1}^{L} P(w_i | w_{i-2} w_{i-1})$$

- Want probabilities to normalize: $\sum_\omega P(\omega) = 1$
- Consider sum of probabilities of one-word sequences.

$$\sum_{w_1} P(\omega = w_1) = \sum_{w_1} p_{\triangleright,\triangleright,w_1} = 1$$

- Fix: introduce end-of-sentence token $w_{L+1} = \triangleleft$

$$P(\omega = w_1 \cdots w_L) = \prod_{i=1}^{L+1} P(w_i | w_{i-2} w_{i-1})$$

In fact, $\sum_{\omega:|\omega|=L} P(\omega) = 1$ for all $L \Rightarrow \sum_\omega P(\omega) = \infty$

Maximum Likelihood Estimation

- Optimize likelihood of each multinomial independently.
  - One multinomial per history.
- ML estimate for multinomials: count and normalize!
- *e.g.*, trigram model:

$$p^{\text{MLE}}_{w_{i-2},w_{i-1},w_i} = \frac{c(w_{i-2} w_{i-1} w_i)}{\sum_w c(w_{i-2} w_{i-1} w)} = \frac{c(w_{i-2} w_{i-1} w_i)}{c(w_{i-2} w_{i-1})}$$

Bigram Model Example

- Training data:

  JOHN READ MOBY DICK
  MARY READ A DIFFERENT BOOK
  SHE READ A BOOK BY CHER

- What is $P($JOHN READ A BOOK$)$?

Bigram Model Example

$$
\begin{aligned}
&P(\textsc{john read a book}) \\
&= P(\textsc{john}|\triangleright)P(\textsc{read}|\textsc{john})P(\textsc{a}|\textsc{read})P(\textsc{book}|\textsc{a})P(\triangleleft|\textsc{book}) \\
&= \frac{1}{3} \times 1 \times \frac{2}{3} \times \frac{1}{2} \times \frac{1}{2} \approx 0.06
\end{aligned}
$$

Recap: *N*-Gram Models

- Simple formalism.
- Easy to train.
  - Just count and normalize.
  - Can train on vast amounts of data; just gets better.

Does Markov Property Hold For English?

- Not for small *n*.

$$P(w_i \mid \textsc{of the}) \neq P(w_i \mid \textsc{king of the})$$

- Make *n* larger?

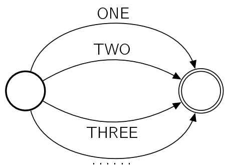
  FABIO, WHO WAS NEXT IN LINE, ASKED IF THE TELLER SPOKE ...

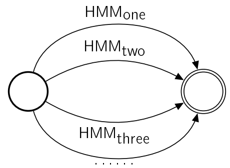
- For vocabulary size $V = 20,000 \ldots$
  - How many parameters (p_{w_{i-1},w_i}) in bigram model?
  - In trigram model?
- Vast majority of trigrams not present in training data!

Where Are We?

LM's and Training and Decoding

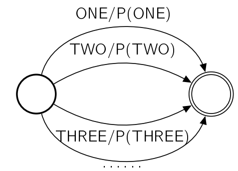
- Decoding without LM's.
  - Start with word HMM encoding allowable word sequences.
  - Replace each word with its HMM.

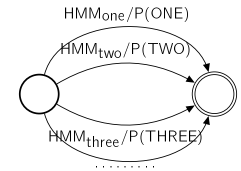




LM's and Training and Decoding

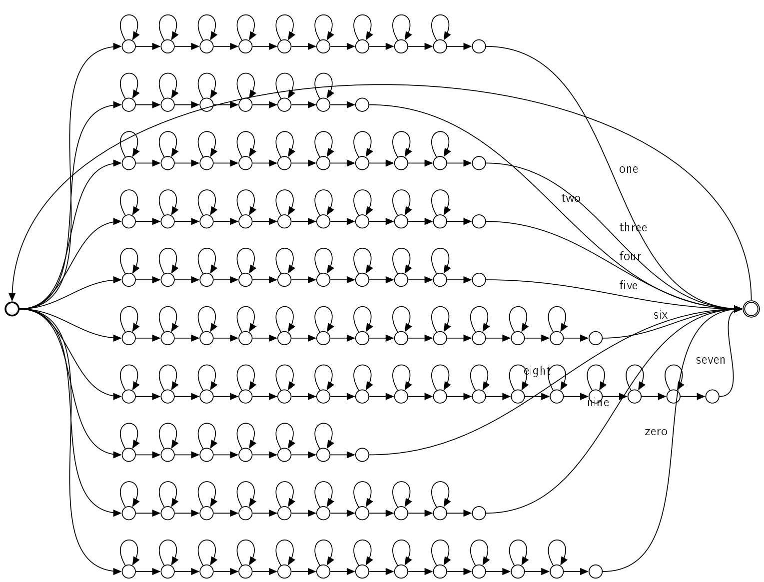
- Point: *n*-gram model is (hidden) Markov model.
  - Can be expressed as word HMM.
  - Replace each word with its HMM.
  - Leave in language model probabilities.





- How do LM's impact acoustic model training?

One Puny Prob versus Many?



The Language Model Weight

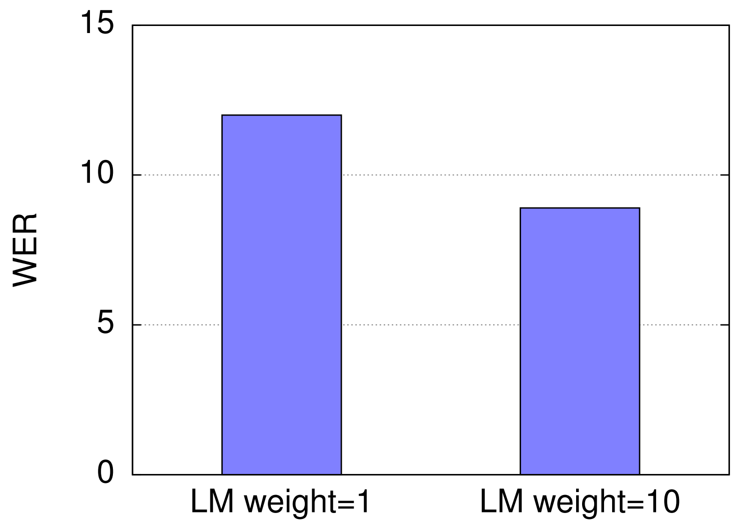
- This doesn't look like fair fight.
- Solution: language (or acoustic) model weight.

$$\omega^* = \arg\max_{\omega} P(\omega)^{\alpha} P(\mathbf{x}|\omega)$$

  - α usually somewhere between 10 and 20.
- Important to tune for each LM, AM.
- Theoretically inelegant.
  - Empirical performance trumps theory any day of week.

Real World Toy Example

- Test set: continuous digit strings.
- *Unigram* language model: $P(\omega) = \prod_{i=1}^{L+1} p_{w_i}$.



What is This Word Error Rate Thing?

- Most popular evaluation measure for ASR systems

$$\text{WER} \equiv \frac{\sum_{\text{utts } u}(\text{\# errors in } u)}{\sum_{\text{utts } u}(\text{\# words in reference for } u)}$$

- \# errors for hypothesis u_{hyp}; reference u_{ref}:
  - Min number of word substitutions, deletions, and ...
  - Insertions to transform u_{ref} into u_{hyp}.
- Example: what is the WER?

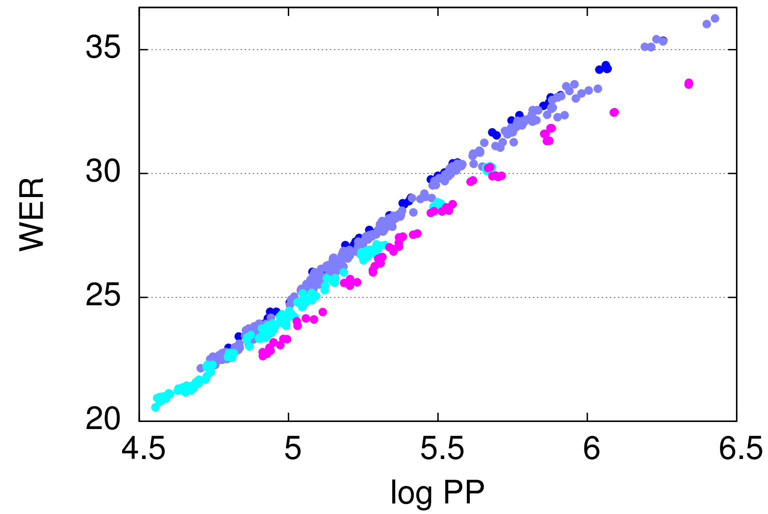
  u_{ref}: THE DOG IS HERE NOW

  u_{hyp}: THE UH BOG IS NOW
- Can WER be above 100%?

Evaluating Language Models

- Best way: plug into ASR system, see how affects WER.
  - Expensive to compute (especially in old days).
  - Results depend on acoustic model.
- Is there something cheaper that predicts WER well?
  - *Perplexity* (PP) of test data (needs only text).
  - Doesn't predict performance well *across* LM types.
  - But does within single LM type!
  - Has theoretical significance.

Perplexity and Word-Error Rate



Perplexity

- Compute (geometric) average probability p_{avg} . . .
  - Assigned to each word in test data.

$$p_{\text{avg}} = \left[\prod_{i=1}^{L} P(w_i | w_{i-2} w_{i-1})\right]^{\frac{1}{L}}$$

- Invert it: $\text{PP} = \frac{1}{p_{\text{avg}}}$
  - Can be interpreted as average branching factor.
- Theoretical significance:
  - $\log_2 \text{PP}$ = average number of bits per word . . .
  - Needed to encode test data using LM.

Perplexity

- Estimate of human performance (Shannon, 1951)
  - Shannon game — humans guess next letter in text.
  - PP=142 (1.3 bits/letter), uncased, unpunctuated.
- Estimate of trigram language model (Brown *et al.*, 1992).
  - PP=790 (1.75 bits/letter), cased, punctuated.
- ASR systems (uncased, unpunctuated, closed vocab).
  - ∼100 for complex domains (*e.g.*, Switchboard, BN).
  - Can be much lower for constrained domains.
  - Can vary widely across languages.

Recap

- LM describes allowable word sequences.
  - Used to build decoding graph.
- Need LM weight for LM to have full effect.
- Best to evaluate LM's using WER . . .
  - But perplexity is informative in some contexts.

Where Are We?

An Experiment

- Take 50M words of WSJ; shuffle sentences; split in two.
- "Training" set: 25M words.

  NONCOMPETITIVE TENDERS MUST BE RECEIVED BY NOON EASTERN TIME THURSDAY AT THE TREASURY OR AT FEDERAL RESERVE BANKS OR BRANCHES .PERIOD
  NOT EVERYONE AGREED WITH THAT STRATEGY .PERIOD
  ...
  ...
- "Test" set: 25M words.

  NATIONAL PICTURE AMPERSAND FRAME –DASH INITIAL TWO MILLION ,COMMA TWO HUNDRED FIFTY THOUSAND SHARES ,COMMA VIA WILLIAM BLAIR .PERIOD
  THERE WILL EVEN BE AN EIGHTEEN -HYPHEN HOLE GOLF COURSE .PERIOD
  ...
  ...

An Experiment

- Count how often each word occurs in training; sort by count.

word	count
,COMMA	1156259
THE	1062057
.PERIOD	877624
OF	520374
TO	510508
A	455832
AND	417364
IN	385940
...	...
...	...

word	count
...	...
...	...
ZZZZ	2
AAAAAHHH	1
AAB	1
AACHENER	1
...	...
...	...
ZYPLAST	1
ZYUGANOV	1

An Experiment

- For each word that occurs exactly once in training ...
  - Count how often occurs in test set.
  - Average this count across all such words.
- What does ML estimate predict?
- What is actual value?
  1. Larger than 1.
  2. Exactly 1, more or less.
  3. Between 0.5 and 1.
  4. Between 0.1 and 0.5.
- What if do this for trigrams, not unigrams?

Why?

- What percentage of words/trigrams in test set ...
  - Had no counts in training set?
  - 0.2%/31%.

Maximum Likelihood and Sparse Data

- In theory, ML estimate is as good as it gets ...
  - In limit of lots of data.
- In practice, sucks when data is *sparse*.
  - Can be off by large factor.

Maximum Likelihood and Zero Probabilities

- According to MLE trigram model ...
  - What is probability of sentence ω if ω contains ...
  - Trigram with no training counts?
- How common are unseen trigrams?
  - (Brown *et al.*, 1992): 350M word training set
  - In test set, what percentage of trigrams unseen?
- How does this affect WER? Perplexity?

Smoothing

- How to adjust ML estimates to better match test data?
- How to avoid zero probabilities?
- Also called *regularization*.

The Basic Idea, Bigram Model

- For each history word w_{i-1} ...
  - Estimate conditional distribution $P(w_i | w_{i-1})$.
- Maximum likelihood estimates.

$$p^{\text{MLE}}_{w_{i-1},w_i} = \frac{c(w_{i-1} w_i)}{c(w_{i-1})}$$

- Give prob to zero counts by discounting nonzero counts.

$$p^{\text{sm}}_{w_{i-1},w_i} = \frac{c(w_{i-1} w_i) - d(w_{i-1} w_i)}{c(w_{i-1})}$$

- How much to discount?

The Good-Turing Estimate

- How often word with k counts in training data ...
  - Occurs in test set of equal size?

$$\text{(avg. count)} \approx \frac{(\text{\# words w/ } k+1 \text{ counts}) \times (k+1)}{(\text{\# words w/ } k \text{ counts})}$$

- How accurate is this?

k	GT estimate	actual
1	0.45	0.45
2	1.26	1.25
3	2.24	2.24
4	3.24	3.23
5	4.22	4.21

The Basic Idea, Bigram Model (cont'd)

- Give prob to zero counts by discounting nonzero counts.
  - Can use GT estimate to determine discounts $d(w_{i-1} w_i)$.

$$p^{\text{sm}}_{w_{i-1},w_i} = \frac{c(w_{i-1} w_i) - d(w_{i-1} w_i)}{c(w_{i-1})}$$

- Total prob freed up for zero counts:

$$P^{\text{sm}}(\text{unseen}|w_{i-1}) = \frac{\sum_{w_i \text{seen}} d(w_{i-1} w_i)}{c(w_{i-1})}$$

- How to divvy up between words unseen after w_{i-1}?

Backoff

- Task: divide up some probability mass ...
  - Among words not occurring after some history w_{i-1}.
- Idea: uniformly?
- Better idea: according to *unigram* distribution.
  - *e.g.*, give more mass to *THE* than *FUGUE*.

$$P(w) = \frac{c(w)}{\sum_w c(w)}$$

- *Backoff*: use lower-order distribution ...
  - To fill in probabilities for unseen words.

Putting It All Together: Katz Smoothing

- Katz (1987)

$$P_{\text{Katz}}(w_i|w_{i-1}) = \begin{cases} P_{\text{MLE}}(w_i|w_{i-1}) & \text{if } c(w_{i-1} w_i) \geq k \\ P_{\text{GT}}(w_i|w_{i-1}) & \text{if } 0 < c(w_{i-1} w_i) < k \\ \alpha_{w_{i-1}} P_{\text{Katz}}(w_i) & \text{otherwise} \end{cases}$$

  - If count high, no discounting (GT estimate unreliable).
  - If count low, use GT estimate.
  - If no count, use scaled backoff probability.
- Choose $\alpha_{w_{i-1}}$ so $\sum_{w_i} P_{\text{Katz}}(w_i|w_{i-1}) = 1$.
- Most popular smoothing technique for about a decade.

Recap: Smoothing

- No smoothing (MLE estimate): performance will suck.
  - Zero probabilities will kill you.
- Key aspects of smoothing algorithms.
  - How to discount counts of seen words.
  - Estimating mass of unseen words.
  - Backoff to get information from lower-order models.
- Lots and lots of smoothing algorithms developed.
  - Will talk about newer algorithms in Lecture 11.
  - Gain: ~1% absolute in WER over Katz.
- No downside to good smoothing (except implementing).

Discussion

- Good smoothing removes performance penalty . . .
  - For overly large models!
- *e.g.*, with lots of data (100MW+) . . .
  - Significant gain for 5-gram model over trigram model.
  - Limiting resource: disk/memory.
- *Count cutoffs* or *entropy-based pruning* . . .
  - Can be used to reduce size of LM.
- Rule of thumb: if ML estimate is working OK . . .
  - Model is way too small.

Where Are We?

N-Gram Models

- Workhorse of language modeling for ASR for 30 years.
  - Used in great majority of deployed systems.
- Almost no linguistic knowledge.
  - Totally data-driven.
- Easy to build.
  - Fast and scalable.

The Fundamental Equation of ASR

$$\omega^* = \arg\max_{\omega} P(\omega|\mathbf{x}) = \arg\max_{\omega} P(\omega)P(\mathbf{x}|\omega)$$

- *Source-channel* model.
  - Source model $P(\omega)$ [language model].
  - (Noisy) channel model $P(\mathbf{x}|\omega)$ [acoustic model].
  - Recover ω despite corruption from noisy channel.
- Many other applications follow same framework.

Where Else Are Language Models Used?

$$\omega^* = \arg\max_{\omega} P(\omega|\mathbf{x}) = \arg\max_{\omega} P(\omega)P(\mathbf{x}|\omega)$$

- Handwriting recognition.
- Optical character recognition.
- Spelling correction.
- Machine translation.
- Natural language generation.
- Information retrieval.
- Any problem involving sequences?

Part III

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)?