

# Lecture 5

## The Big Picture/Language Modeling

Bhuvana Ramabhadran, Michael Picheny, Stanley F. Chen

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

06 October 2009



# Administrivia

- Feedback from last lecture.
  - More on the big picture?
  - Be careful what you wish for.
- Lab 1
  - Not graded yet; will be graded by next lecture.
  - Awards ceremony for evaluation next week.
- Lab 2
  - Handed out last Saturday; due nine days from now (Thursday, Oct. 15) at midnight.
  - Start early! Avail yourself of Courseworks.



# Outline

- The Big Picture (review).
  - The story so far, plus situating Lab 1 and Lab 2.
- Language modeling.
  - Why?
  - How? (Grammars,  $n$ -gram models.)
  - Parameter estimation (smoothing).
  - Evaluation.



# Part I

## The Big Picture



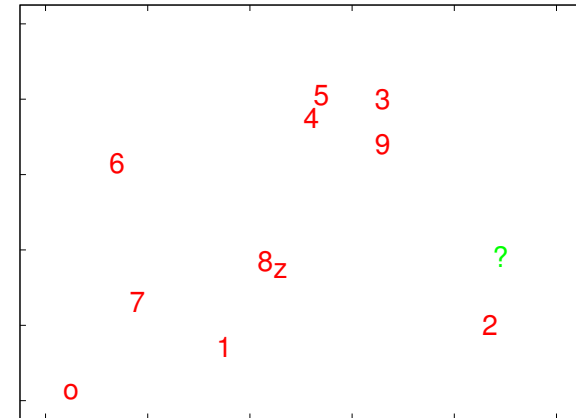
# Where Are We?

- 1 DTW as Nearest Neighbor
- 2 Probabilistic Modeling
- 3 From DTW To Probabilities
- 4 How Big Should Models Be?
- 5 Training and Decoding Logistics



# DTW as Nearest Neighbor Classification

- One template/exemplar per class.
- Find nearest class.



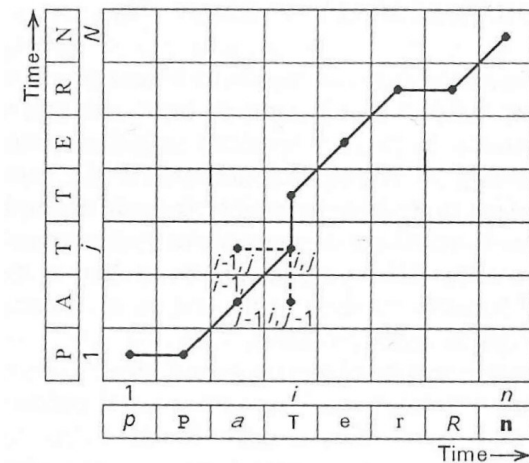
# DTW as Nearest Neighbor Classification

- In reality ...
  - Each example is an audio signal of arbitrary length/dimension.
- The key question ...
  - What's a good distance measure/feature space?



# Problem 1: Varying Utterance Lengths

- An *alignment* maps each test frame to a template frame.
- Sum distances between aligned frames.



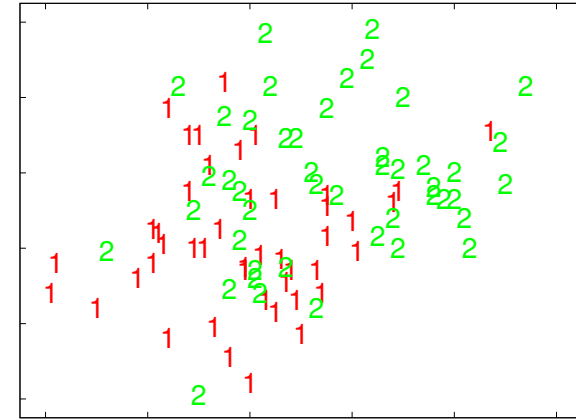
## Dynamic Time Warping

- Which alignment to use?
  - Pick alignment that results in lowest overall distance.
- How to find this alignment efficiently?
  - *Dynamic programming* lets us search exponential number of alignments in polynomial time.
- Penalize “weird” alignments.
  - Add alignment cost into overall distance to penalize skips and the such.



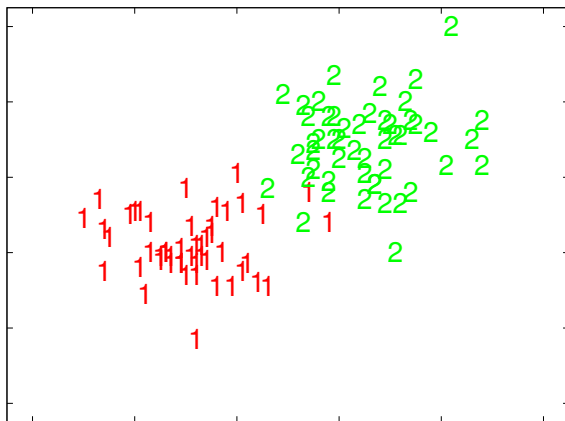
## Problem 2: Computing Frame Distances

- With bad feature representation, classification is hard.



## Problem 2: Computing Frame Distances

- With good feature representation, classification is easy.



## What Did We Learn in Lab 1?

- Goal: maximize separability between examples from different classes.
- A smoothed frequency-domain representation works well.
  - FFT computes energy at each frequency (fine-grained).
  - Mel-binning does coarse-grained grouping (93.6% accuracy).
  - Log+DCT, Hamming window help.
  - Not completely unlike human auditory system.
- Data reduction makes later processing go faster.
  - Instead of processing 20000 values/sec, process 100×12 values/sec.



## What Did We Learn in Lab 1?

- DTW is OK for speaker-dependent small-voc isolated word ASR.
  - >95% accuracy in clean conditions.
- Can we do better?



## The Key: A Good Distance Measure

- Can we improve upon our *ad hoc* distance measure?
- There are lots of parameters that one can tune, e.g.,
  - dimension-dependent weights
  - class-dependent weights
  - alignment costs
  - relative weight of frame distances and alignment costs
  - etc.
- Is there a framework for choosing these parameters in a principled manner?



## Where Are We?

- 1 DTW as Nearest Neighbor
- 2 Probabilistic Modeling
- 3 From DTW To Probabilities
- 4 How Big Should Models Be?
- 5 Training and Decoding Logistics



## The Idea

- Old way: choose class that minimizes
$$\text{distance}(\text{class}, \text{example})$$
- New way: choose class that maximizes
$$P(\text{example}|\text{class})$$



## Probabilities as Frequencies

- What do we mean by  $P(\text{example}|\text{class})$ ?
  - The (relative) frequency with which the word *class* ...
  - Is realized as the feature vector *example*.



## What Does Probabilistic Modeling Give Us?

- If we can estimate  $P(\text{example}|\text{class})$  perfectly ...
  - We can perform classification optimally!
- e.g., two-class classification (w/ classes equally frequent)
  - Choose YES if  $P(\text{example}|\text{YES}) > P(\text{example}|\text{NO})$ .
  - Choose NO otherwise.
  - This is the best you can do!



## Can We Find “Perfect” Models?

- Models have *parameters*, e.g., consider a Gaussian.

$$P(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp \left[ -\frac{1}{2} \left( \frac{x - \mu}{\sigma} \right)^2 \right]$$

- Parameters:  $\mu, \sigma$ .
- Idea: *Maximum likelihood estimation* (MLE).
  - Choose parameters that maximize the likelihood of the training data  $\{x_i\}$ :

$$(\mu_{\text{MLE}}, \sigma_{\text{MLE}}) = \arg \max_{(\mu, \sigma)} \prod_i P(x_i|\mu, \sigma)$$



## Maximum Likelihood Estimation

- Assume we have the “correct” model form.
- Then, in the presence of infinite training samples ...
  - Maximum likelihood estimation (MLE) approaches the “true” parameter estimates.
  - For most models, MLE is asymptotically consistent, unbiased, and efficient.
- Maximum likelihood estimation is easy for a wide class of models.
  - $\Rightarrow$  Count and normalize!



# Long Live Probabilistic Modeling!

- If we choose the form of our model correctly and have lots of data ...
  - We win!
- ASR moved to probabilistic modeling in mid-70's.
  - In ASR, no serious challenges to this framework since then.
- By and large, probabilistic modeling ...
  - Does as well or better than all other techniques ...
  - On all classification problems.
  - (Not everyone agrees.)



# What's the Catch?

- Do we know form of probability distribution that data comes from?
- Can we actually find the MLE?
  - In training HMM's w/ FB, can only find local optimum.
  - MLE solution for GMM's is pathological.
- Do we have enough training data?
  - "There's no data like more data."
- These are the challenges!



# Where Are We?

- 1 DTW as Nearest Neighbor
- 2 Probabilistic Modeling
- 3 From DTW To Probabilities
- 4 How Big Should Models Be?
- 5 Training and Decoding Logistics



# A Probabilistic Model of Speech

- Isolated word recognition w/ front end processing.

$$P(\text{feature vectors} \mid \text{word})$$

- Notation:  $P(\mathbf{x}|\omega) \Rightarrow P_\omega(\mathbf{x})$



## Problem 1: Varying Utterance Lengths

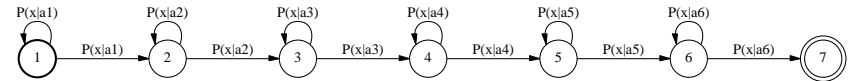
- An *alignment*  $A = a_1 \cdots a_t$  maps each test frame  $t$  to a template frame  $a_t$ .
  - To compute probability of feature vectors  $\mathbf{x} \dots$
  - Sum over all possible alignments  $A$ .

$$\begin{aligned} P_\omega(\mathbf{x}) &= \sum_A P_\omega(\mathbf{x}, A) \\ &= \sum_A P_\omega(A) \times P_\omega(\mathbf{x}|A) \end{aligned}$$



## A Model of Alignments

- How to model  $P_\omega(A)$ ?
  - HMM's!?
- Replace template for each word with an HMM.
  - Each frame in template corresponds to state in HMM.
  - Each alignment corresponds to path through HMM ...
  - Mapping each frame  $t$  to arc  $a_t$  in HMM.



- Path probability is product of transition probabilities along path.

$$P_\omega(A) = \prod_{t=1}^T P(a_t)$$



## Problem 2: Computing Frame Distances

$$\begin{aligned} P_\omega(\mathbf{x}) &= \sum_A P_\omega(\mathbf{x}, A) \\ &= \sum_A P_\omega(A) \times P_\omega(\mathbf{x}|A) \end{aligned}$$

- How to model  $P_\omega(\mathbf{x}|A)$ ?
- Compute overall probability by combining probability of each frame.

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



## Problem 2: Computing Frame Distances

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

- How to model frame probabilities?
  - GMM!?
- Lots of stuff is Gaussian.
  - If you sum enough Gaussians ...
  - You can approximate any distribution arbitrarily well.



## Putting It All Together

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



## DTW and HMM's

- Can design HMM such that (see Holmes, Sec. 9.13, p. 155).

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

| DTW                     | HMM                       |
|-------------------------|---------------------------|
| acoustic template       | HMM                       |
| frame in template       | state in HMM              |
| DTW alignment           | path through HMM          |
| move cost               | transition (log)prob      |
| distance between frames | output (log)prob          |
| DTW search              | Forward/Viterbi algorithm |



## What Have We Gained?

- Automatically estimate parameters from data, scalably.
  - Maximum likelihood estimates improve with more data.
- Easy to encompass rich alignment and feature models.
  - e.g., Gaussian variances  $\Rightarrow$  class- and dimension-dependent distances.



## Point to Ponder

- What independence assumptions have we made?



# Where Are We?

- 1 DTW as Nearest Neighbor
- 2 Probabilistic Modeling
- 3 From DTW To Probabilities
- 4 How Big Should Models Be?
- 5 Training and Decoding Logistics



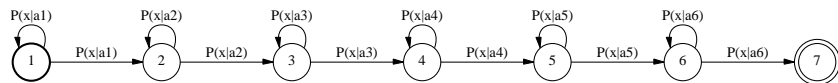
# Economizing on Parameters

- There is a cost to having too many parameters.
  - Too little data per parameter  $\Rightarrow$  poor estimation.
  - Computational cost.
  - Local minima in EM training.
- Try to model the important stuff with as few parameters as possible.



# How Big Should an HMM Be?

- Original view: each frame in DTW template for word  $\omega$  corresponds to state in HMM.
  - But consecutive frames may be very similar to each other!

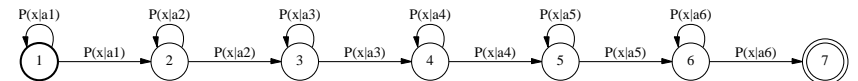


- How many states do we really need?
  - Rule of thumb: three states per *phoneme*.



# Word HMM's

- Word TWO is composed of phonemes T UW.
- Two phonemes  $\Rightarrow$  six HMM states.



- Convention: same output distribution for all arcs exiting a state ...
  - Or place output distributions on states.
- Parameters: 12 transition probabilities, 6 output distributions.



## How Big Should a GMM Be?

- For a Gaussian over  $d \approx 40$  dimensions:
  - Diagonal covariance:  $d + d$  parameters.
  - Full covariance:  $d + \frac{d(d+1)}{2}$  parameters.
- How many components in each GMM?
  - Maybe 20–40, depending on how much data you have.



## Where Are We?

- 1 DTW as Nearest Neighbor
- 2 Probabilistic Modeling
- 3 From DTW To Probabilities
- 4 How Big Should Models Be?
- 5 Training and Decoding Logistics



## Training for Isolated Word Recognition

- What do you need?
  - A set of audio utterances.
  - The reference transcript for each utterance.
  - An HMM topology for each word.



## Training: The Basic Idea

- For each class (word)  $\omega \dots$ 
  - Collect all training samples  $\mathbf{x}$  with that label.
  - Run FB on these samples to train parameters of corresponding HMM/GMM's.



## The Way Things Actually Work

- Initialize training counts to 0.
- For each utterance  $(\mathbf{x}_i, \omega_i)$  in training data ...
  - Calculate acoustic feature vectors  $\mathbf{x}_i$  using front end.
  - Construct HMM corresponding to word  $\omega_i$ .
  - Run FB algorithm to update training counts for all transitions, GMM's in HMM.
- Reestimate HMM parameters using training counts.
- Repeat until training likelihood  $\prod_i P_{\omega_i}(\mathbf{x}_i)$  converges.



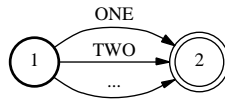
## Decoding: The Basic Idea

- For a test example  $\mathbf{x}_{\text{test}}$ .
  - For each word  $\omega$ , compute (Viterbi or Forward) likelihood  $P_{\omega}(\mathbf{x}_{\text{test}})$  using word HMM.
  - Pick word producing highest probability.
- Forward algorithm:  $P_{\omega}(\mathbf{x}) = \sum_A P_{\omega}(\mathbf{x}, A)$
- Viterbi algorithm:  $P_{\omega}(\mathbf{x}) \approx \max_A P_{\omega}(\mathbf{x}, A)$



## The Way Things Actually Work

- Instead of separate HMM for each word  $\omega$  ...
  - Merge each word HMM into single big HMM.



- Use Viterbi algorithm (not Forward algorithm, why?)
  - In backtrace, collect all words along the way.
- Why one big graph?
  - Scalability to word sequences.
  - Speed: pruning during search; graph minimization.



## From Isolated to Continuous ASR

- e.g., ASR for digit sequences of arbitrary length.
- The key insight:

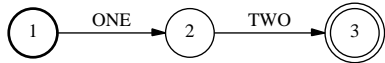
$$P_{\omega_1 \omega_2}(\mathbf{x}_1 \mathbf{x}_2) = P_{\omega_1}(\mathbf{x}_1) \times P_{\omega_2}(\mathbf{x}_2)$$

- Can construct HMM for a word sequence ...
  - By concatenating component word HMM's.
- Can use same exact word HMM's and HMM parameters as before!

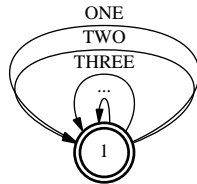


# Continuous Speech Recognition with HMM's

- Training: same as before, except ...
  - HMM for utterance constructed by concatenating word HMM's for reference transcript.



- Decoding: same as before, except ...
  - The single big HMM must include paths for all word sequences under consideration.



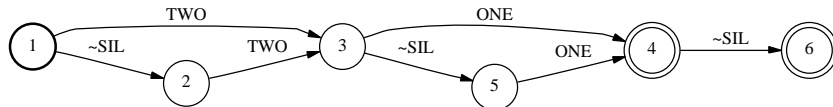
# Silence is Golden

- What about all those pauses in speech?
  - At begin/end of utterance; in between words.
- Add silence word ~SIL to vocabulary.
  - Three states? (Skip arcs?)
- Allow optional silence everywhere.



# Silence is Golden

- e.g., HMM for training with reference script: ONE TWO



- Lab 2: all these graphs are automatically constructed for you.
  - Silence is also used in isolated word training/decoding.



# Part II

## Language Modeling



## Wreck a Nice Beach?

- The current story: *maximum likelihood classification*.

- To decode a test sample  $\mathbf{x}_{\text{test}}$  ...
- Find word sequence  $\omega$  maximizing likelihood

$$P_{\omega}(\mathbf{x}_{\text{test}}|\omega).$$

- But:

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



## The Fundamental Equation of ASR

$$\text{class}(\mathbf{x}_{\text{test}}) \stackrel{?}{=} \arg \max_{\omega} P(\mathbf{x}_{\text{test}}|\omega)$$

$$\text{class}(\mathbf{x}_{\text{test}}) \stackrel{!}{=} \arg \max_{\omega} P(\omega|\mathbf{x}_{\text{test}})$$



## The Fundamental Equation of ASR

- Bayes' rule:

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

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

- The fundamental equation:

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



## The Fundamental Equation of ASR

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

- $P(\omega)$  = *language model*.
  - Models frequency of each word sequence  $\omega$ .
- $P(\mathbf{x}|\omega)$  = *acoustic model*.
  - Models frequency of acoustic feature vectors  $\mathbf{x}$  given word sequence  $\omega$ .

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



## Source-Channel Modeling

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

- *Source-channel model.*
  - Source model  $P(\omega)$  [language model].
  - (Noisy) channel model  $P(\mathbf{x}|\omega)$  [acoustic model].
  - Recover source  $\omega$  despite corruption from noisy channel.
- *Channel model a.k.a. direct model.*
  - Model  $P(\omega|\mathbf{x})$  directly.
  - Why not?



## Where Else Are Language Models Used?

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



## Language Modeling and Domain

- Isolated digits: implicit language model.

$$P(\text{ONE}) = \frac{1}{11}, P(\text{TWO}) = \frac{1}{11}, \dots, P(\text{ZERO}) = \frac{1}{11}, P(\text{OH}) = \frac{1}{11}$$

- All other word sequences have probability zero.
- Language models describe what word sequences the domain allow.
- The better you model acceptable/likely word sequences ...
- The fewer acceptable/likely word sequences in a domain ...
  - If not many choices, bad acoustic models look OK.
  - e.g., isolated digit recognition, YES/NO recognition.



## Real World Examples

- Isolated digits test set (*i.e.*, single digits).
- Language model 1.
  - Each digit sequence of length 1 equiprobable.
  - Probability zero for all other digit sequences.
- Language model 2.
  - Each digit sequence (of any length) equiprobable.
- LM 1: 1.8% error rate; LM 2: 11.5% error rate.
- Point: use all available domain knowledge in your LM!
  - e.g., name dialer, phone number entry, UPS tracking number.



# How to Construct a Language Model

For really simple domains:

- Enumerate all allowable word sequences.
  - *i.e.*, all word sequences  $\omega$  with  $P(\omega) > 0$ .
  - *e.g.*, YES/NO, isolated digits.
- Use common sense to set  $P(\omega)$ .
  - Uniform distribution:  $P(\omega) = \frac{1}{\text{size}(\{\omega\})}$ .
  - Reduces to maximum likelihood classification:

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



# How to Construct a Language Model

For pretty simple domains:

- *e.g.*, 7-digit phone numbers.
  - OH OH OH OH OH OH OH
  - OH OH OH OH OH OH ONE
  - OH OH OH OH OH OH TWO
  - etc.
- Is there a way we can compactly represent this list of strings?



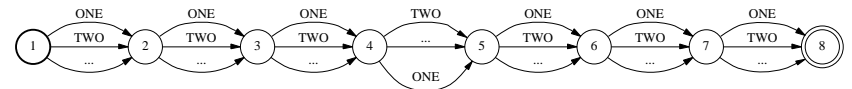
## Where Are We?

- 6 Grammatical LM's
- 7 N-Gram Models
- 8 Smoothing
- 9 Evaluating Language Models

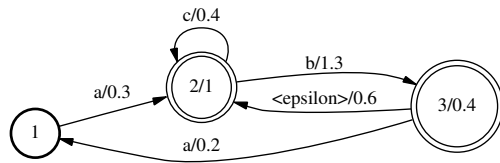


## Finite-State Automata

- What about (Hidden) Markov models?
- Let's use a different name.
  - Finite-state automaton/machine (FSA/FSM).
  - Or *weighted* finite-state automaton/machine (WFSA/WFSM).
  - Or *grammar*.



# Weighted Finite-State Automata



## Conventions:

- Single start state.
- Arcs are labeled with a word and have a probability.
- Can have multiple final states; final probability.



# WFSA's as Language Models

- Probability of path (same as HMM):
  - Product of probabilities of each arc along path times ...
  - Final probability of state at end of path.

$$P(\text{path } A = a_1 \dots a_T) = \prod_{t=1}^T P(a_t) \times P_{\text{final}}(\text{dest}(a_T))$$

- Probability of a word string  $\omega$  (same as HMM):
  - Sum over all paths labeled with that word string.

$$P(\omega) = \sum_{A: \text{output}(A)=\omega} P(A)$$



# Building a Grammar LM

- Step 1: Collect some in-domain training data.
  - Just need text, not audio.
- For some domains, easy:
  - News.
  - Data sources: the Web, Linguistic Data Consortium.
- For some domains, hard:
  - Telephone conversations; dialogue systems.
  - May need to pay people to collect/generate data.



# Constructing WFSA's

- The usual way.
  - Build WFSA topology manually ...
  - So that "accepts" most of the training data.
  - Set probabilities manually.
- Another possible way.
  - Learn WFSA topology automatically from training data.
  - Train transition probabilities automatically. (How?)



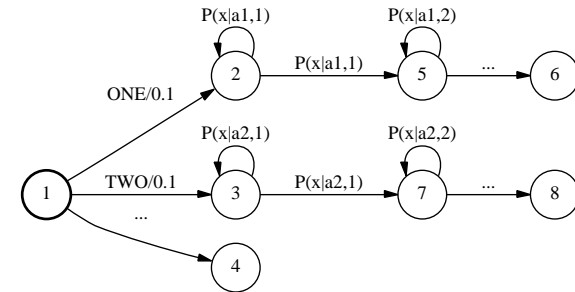
# LM's and Training and Decoding

- How do LM's impact acoustic model training?
- How do LM's impact decoding?
  - Old way: build big HMM by taking FSA representing all allowable word sequences ...
  - And replace each word with its HMM.



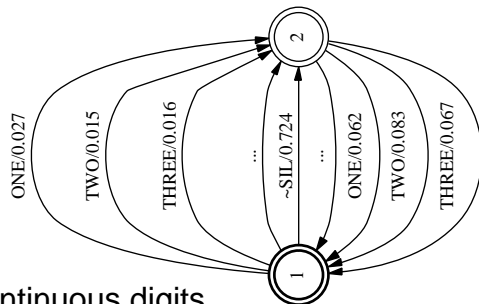
# Using a Grammar LM in Decoding

- How do LM's impact decoding?
  - New way: build big HMM by taking *weighted* FSA representing all allowable word sequences ...
  - And replace each word with its HMM.
- e.g., digit grammar



# Real World Example

- Decode continuous digit training data (4300 utts).
  - Why not just take reference transcript?
- Run Forward-Backward algorithm with following topology:



- Decoding continuous digits.
  - w/o LM: 13.1% WER (word error rate)
  - w/ LM: 12.0% WER
  - w/ LM, LM weight=10: 8.9% WER



# Aside: What's Up With That LM Weight?

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

- Or is it the acoustic model weight?
- $\alpha$  often somewhere between 10 and 20
- One theory: modeling error.
  - If we could estimate  $P(\omega)$  and  $P(\mathbf{x}|\omega)$  perfectly ...
  - e.g., at a given arc  $a_t$ , acoustic model assumes frames are independent

$$P(\vec{x}_t, \vec{x}_{t+1} | a_t = a_{t+1}) = P(\vec{x}_t | a_t) P(\vec{x}_{t+1} | a_t)$$



## Aside: What's Up With That LM Weight?

- Another theory: higher variance in estimates of acoustic model probs.
  - Generally,  $|\log P(\mathbf{x}|\omega)| \gg |\log P(\omega)|$
  - $\log P(\mathbf{x}|\omega)$  is computed by summing many more terms.
  - e.g., continuous digits,  $|\log P(\mathbf{x}|\omega)| \sim 1000$ ,  
 $|\log P(\omega)| \sim 20$
  - Variance is cumulative.
  - Smaller differences in LM logprobs are more statistically significant.



## Aside: 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 hypothesized word sequence  $u_{\text{hyp}}$ ; reference  $u_{\text{ref}}$ 
  - Minimum number of word substitutions, word deletions, and word insertions needed to transform  $u_{\text{ref}}$  into  $u_{\text{hyp}}$
- Example:  $\text{WER} = \frac{3}{5} = 60\%$

$u_{\text{ref}}$ : THE DOG IS HERE NOW  
 $u_{\text{hyp}}$ : THE UH BOG IS NOW

- Can WER be above 100%?



## Aside: What is This Word Error Rate Thing?

- How can we compute the number of errors efficiently?
- Technical note: normalize word sequences beforehand.
  - e.g., remove ~SIL; change OH to ZERO
- *word accuracy*: percentage of reference words correct (ignores insertions).
- *sentence error rate*: percentage of utterances with at least one error.
- Not all errors are made the same.
  - e.g., article vs. digit in voicemail.



## Grammars Redux

- Is there a nicer syntax for specifying FSM's?
  - Awkward to type in FSM's.
  - e.g., "arc from state 3 to state 6 with label SEVEN"
- Backus-Naur Form (BNF)

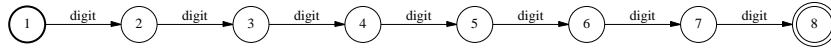
[noun phrase] → [determiner] [noun]  
[determiner] → A | THE  
[noun] → CAT | DOG

- How to express 7-digit phone numbers in BNF?

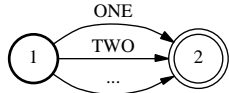


## Compiling a BNF Grammar Into an FSM

- Express each individual rule as an FSM.  
phone-number



digit



- Replace symbol with its FSM.
- Can we handle recursion/self-reference?



## Aside: The Chomsky Hierarchy

- An FSA encodes a set of word sequences.
- A set of word sequences is called a *language*.
- Chomsky hierarchy.
  - Regular language: a language expressible by (finite) FSA.
  - Context-free language: a language expressible in BNF.
  - $\{\text{regular languages}\} \subset \{\text{context-free languages}\}$
  - e.g., the language  $a^n b^n$  for  $n = 1, 2, \dots$



## Aside: The Chomsky Hierarchy

- Is English regular? (*i.e.*, can it be expressed with an FSA?)
  - Probably not.
- Is English context-free? Are all natural languages context-free?
  - Swiss German, Bambara (Mali)
- Well, why don't we just write down a grammar for English?
  - We're too stupid.
  - Especially since people don't follow the rules.
  - Machine learning can't do it either.



## When Grammars Just Won't Do

- Can't write or induce grammars for complex domains.
  - What to do?
- Can we come up with some sort of a grammar that:
  - Has a "regular" topology that we can just write down.
  - Has lots of parameters that can capture the most important dependencies in language.
  - But not too many, so we can estimate them using not too much data.



## Where Are We?

- 6 Grammatical LM's
- 7 **N-Gram Models**
- 8 Smoothing
- 9 Evaluating Language Models



## N-Gram Models

$$\begin{aligned} P(\omega = w_1 \cdots w_l) &= P(w_1)P(w_2|w_1)P(w_3|w_1 w_2) \cdots P(w_l|w_1 \cdots w_{l-1}) \\ &= \prod_{i=1}^l P(w_i|w_1 \cdots w_{i-1}) \end{aligned}$$

- Markov assumption: identity of next word depends only on last  $n - 1$  words, say  $n=3$

$$P(w_i|w_1 \cdots w_{i-1}) \approx P(w_i|w_{i-2}w_{i-1})$$

- Markov  $\Rightarrow$  Markov Model  $\Rightarrow$  grammar.



## An LM Only an Engineer Could Love

- Markov assumption is clearly false.

$$P(w_i | \text{OF THE}) \Leftrightarrow P(w_i | \text{KING OF THE})$$

- Make  $n$  larger?

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

- Too many parameters.



## N-Gram Models

- Anyhoo, estimating  $P(w_i|w_{i-2}w_{i-1})$  (MLE)

$$\begin{aligned} P_{\text{MLE}}(w_i|w_{i-2}w_{i-1}) &= \frac{\text{count}(w_{i-2}w_{i-1}w_i)}{\sum_{w_i} \text{count}(w_{i-2}w_{i-1}w_i)} \\ &= \frac{\text{count}(w_{i-2}w_{i-1}w_i)}{\text{count}(w_{i-2}w_{i-1})} \end{aligned}$$

- With large training set, we see a reasonable fraction of all likely trigrams (or do we?)



## N-Gram Models

- The workhorse of language modeling for ASR.
  - Around for 30 years; still primary technology for LVCSR.
  - Uses almost no linguistic knowledge.  
*every time i fire a linguist the performance of the recognizer improves.* Fred Jelinek (1988, IBM)
- Do we see a pattern here?



## Technical Details: 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:  $w_{-1} = w_0 = \triangleright$ .



## Technical Details: 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.

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

- Indeed,  $\sum_{\omega:|\omega|=n} P(\omega) = 1$  for all  $n \Rightarrow \sum_{\omega} P(\omega) = \infty$
- Fix: introduce an end-of-sentence token  $w_{n+1} = \triangleleft$

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



## Bigram Model Example

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

$$P(\text{JOHN}|\triangleright) = \frac{\text{count}(\triangleright \text{JOHN})}{\sum_w \text{count}(\triangleright w)} = \frac{1}{3}$$

$$P(\text{READ}|\text{JOHN}) = \frac{\text{count}(\text{JOHN READ})}{\sum_w \text{count}(\text{JOHN } w)} = \frac{1}{1}$$

$$P(\text{A}|\text{READ}) = \frac{\text{count}(\text{READ A})}{\sum_w \text{count}(\text{READ } w)} = \frac{2}{3}$$

$$P(\text{BOOK}|\text{A}) = \frac{\text{count}(\text{A BOOK})}{\sum_w \text{count}(\text{A } w)} = \frac{1}{2}$$

$$P(\triangleleft|\text{BOOK}) = \frac{\text{count}(\text{BOOK } \triangleleft)}{\sum_w \text{count}(\text{BOOK } w)} = \frac{1}{2}$$



## Bigram Model Example

$$\begin{aligned} P(\text{JOHN READ A BOOK}) \\ &= P(\text{JOHN}|\triangleright)P(\text{READ}|\text{JOHN})P(\text{A}|\text{READ})P(\text{BOOK}|\text{A})P(\triangleleft|\text{BOOK}) \\ &= \frac{1}{3} \times 1 \times \frac{2}{3} \times \frac{1}{2} \times \frac{1}{2} \approx 0.06 \end{aligned}$$



## Data Sparsity

Will we see all the trigrams we need to see?

- (Brown *et al.*, 1992): 350M word training set
  - In test set, what percentage of trigrams unseen? > 15%
- *i.e.*, In eight-word sentence, one unseen trigram on average

$$P(w_i | w_{i-2} w_{i-1}) = \frac{\text{count}(w_{i-2} w_{i-1} w_i)}{\sum_{w_i} \text{count}(w_{i-2} w_{i-1} w_i)}$$

- One unseen trigram  $\Rightarrow P(\omega = w_1 \dots w_l) = 0!$
- Decoders prefer word sequences with nonzero probability.



## Life After Maximum Likelihood

Maybe MLE isn't such a great idea after all?

- Experiment: split 44M word data set into two halves (Church and Gale, 1992)
- For a bigram that occurs, say, 5 times in first half ...
  - How many times does it occur in second half on average?
  - MLE predicts 5
  - In reality:  $\sim 4.21 \Rightarrow$  huh?
- What up?



## Where Are We?

- 6 Grammatical LM's
- 7 N-Gram Models
- 8 Smoothing
- 9 Evaluating Language Models



# Bayesian Estimation

- Let's say I take a coin out of my pocket, flip it, and it's "heads".
  - What is  $p = P(\text{heads})$ ?
  - MLE:  $p_{\text{MLE}} = \arg \max_p P(\text{data } D | p) = \frac{1}{1} = 1$ .
  - In reality, we believe  $p \approx 0.5$ .
- Instead of maximize probability of data given  $p$  . . . .
  - Maximize probability of  $p$  given data:  $P(p | \text{data } D)$ .
  - *Maximum a posteriori* (MAP) estimate.



# Bayesian Estimation

$$\begin{aligned} p_{\text{MAP}} &= \arg \max_p P(p | D) \\ &= \arg \max_p \frac{P(p)P(D | p)}{P(D)} \\ &= \arg \max_p P(p)P(D | p) \end{aligned}$$

- $P(p)$  — *prior* probability of coin having a certain  $p$
- $P(D | p)$  — *likelihood* of data given  $p$
- $P(p | D)$  — *posterior* probability of a  $p$  value given the data



# Bayesian Estimation

- prior distribution:  $P(p = 0.5) = 0.99$   
 $P(p = 0.000) = P(p = 0.001) = \dots = P(p = 1.000) = 0.00001$
- data  $D$ : 1 flip, 1 heads
  - $P(p = 0.5 | D) \propto P(p = 0.5)P(D | p = 0.5) = 0.99 \times 0.5 = 0.495$
  - $P(p = 1.0 | D) \propto P(p = 1)P(D | p = 1) = 0.00001 \times 1 = 0.00001$
  - MAP estimate:  $p = 0.5$
- data  $D$ : 17 flips, 17 heads
  - $P(p = 0.5 | D) \propto 0.99 \times 0.5^{17} \approx 0.000008$
  - $P(p = 1.0 | D) \propto 0.00001 \times 1 = 0.00001$
  - MAP estimate:  $p = 1$



# Bayesian Estimation

- Little data, prior has a big effect.
- Lots of data, prior has little effect.
  - MAP estimate converges to ML estimate.



## Language Model Smoothing

- How can we adjust maximum likelihood estimates ...
  - To account for effects of prior distribution ...
  - Especially in the presence of sparse data.
- Usually, don't actually come up with explicit prior.
  - Just use as justification for *ad hoc* methods.



## Example of Language Model Smoothing

+ $\delta$  smoothing (Dirichlet prior)

- add  $\delta$  to each trigram count, for some  $\delta$  (e.g., 0.01)
- $\text{count}(\text{ATE TWO DUCKS}) = 1$ ,  $\text{count}(\text{ATE TWO}) = 1$

$$P_{+\delta}(\text{DUCKS} \mid \text{ATE TWO}) = \frac{1 + 0.01}{1 + 0.01 \times |V|}$$

- vocabulary  $V$  — set of all words under consideration
  - say  $|V| = 10000$

$$P_{+\delta}(\text{DUCKS} \mid \text{ATE TWO}) = \frac{1 + 0.01}{1 + 0.01 \times 10000} = 0.01$$



## Vocabulary Selection

- Tradeoff
  - The more words, the more things you can confuse the correct words with.
  - The fewer words, the more out-of-vocabulary (OOV) words.
  - You can't get a word correct if it's not in the vocabulary.
- In practice:
  - Just choose  $k$  most frequent words in training data.
  - $k \approx 50000$  for unconstrained English speech.
  - $k < 10000$  for constrained tasks.



## Schemes for Language Model Smoothing

- Basic idea: if not enough counts in trigram (bigram) model:
  - Use information from bigram (unigram) model
  - $P(w \mid \text{ATE TWO})$ ,  $P(w \mid \text{TWO})$ ,  $P(w)$
- Backoff (Katz, 1987)
  - Use MLE if enough counts, otherwise *backoff* to lower-order model

$$P_{\text{Katz}}(w_i \mid w_{i-1}) = \begin{cases} P_{\text{MLE}}(w_i \mid w_{i-1}) & \text{if } \text{count}(w_{i-1} w_i) \geq 5 \\ P_{\text{GT}}(w_i \mid w_{i-1}) & \text{if } 0 < \text{count}(w_{i-1} w_i) < 5 \\ \alpha_{w_{i-1}} P_{\text{Katz}}(w_i) & \text{otherwise} \end{cases}$$

- Choose  $\alpha_{w_{i-1}}$  so that  $\sum_{w_i} P_{\text{Katz}}(w_i \mid w_{i-1}) = 1$



## Schemes for Language Model Smoothing

- Interpolation (Jelinek, 1980)
  - Interpolate (weighted sum) with lower-order model.
$$P_{\text{interp}}(w_i | w_{i-1}) = \lambda_{w_{i-1}} P_{\text{MLE}}(w_i | w_{i-1}) + (1 - \lambda_{w_{i-1}}) P_{\text{interp}}(w_i)$$
  - $\lambda_{w_{i-1}}$  depends on count of  $w_{i-1}$ 
    - e.g., if  $\text{count}(w_{i-1}) = 0$ ,  $\lambda_{w_{i-1}} = 0$
    - e.g., if  $\text{count}(w_{i-1}) \gg 0$ ,  $\lambda_{w_{i-1}} \sim 1$
  - Can estimate  $\lambda_{w_{i-1}}$  to maximize likelihood of *held-out* data.
- Empirically, interpolation is superior to backoff.
- State-of-the-art: modified Kneser-Ney smoothing (Chen and Goodman, 1999).



## Optimizing Smoothing Parameters

- Smoothing: modeling difference between training data and “true” distribution (test data).
  - e.g., in training, see  $P_{\text{MLE}}(\text{DUCKS} | \text{ATE TWO}) = \frac{1}{1} = 1$
  - In test, estimate  $P_{\text{smooth}}(\text{DUCKS} | \text{ATE TWO}) \approx 0.2$
- Optimize smoothing parameters on *held-out* data.
$$P_{\text{smooth}}(w_i | w_{i-1}) = \lambda_{w_{i-1}} P_{\text{MLE}}(w_i | w_{i-1}) + (1 - \lambda_{w_{i-1}}) P_{\text{smooth}}(w_i)$$
- What if we chose  $\lambda_{w_{i-1}}$  to optimize likelihood of *training* data?
- Hint: what does “MLE” mean?



## Deleted Estimation

- *Held-out estimation*
  - Reserve part of data as held-out.
  - Can we not “waste” data?
- *Deleted estimation*
  - Reserve part of data as held-out.
  - Rotate!
  - Collect counts from all different partitionings to estimate parameters.



## Good-Turing Discounting

- Estimate how often word with  $k$  counts in training data ...
  - Occurs in test set of equal size.
- Deleted estimation.
  - Delete one word at a time.
  - If “test” word occurs  $k + 1$  times in whole data set ...
    - Occurs  $k$  times in “training” set.
    - Give count to bucket for words with  $k$  counts.
  - Total count placed in bucket for  $k$ -count words

$$(\# \text{ words w/ } k + 1 \text{ counts}) \times (k + 1)$$



## Good-Turing Discounting

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

- 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   |

- Less accurate for higher  $k$ ; why?
- Corollary: (“true” count of unseen events) = (# 1-counts)
  - 53M trigrams w/ 1-count in 365MW of data  $\Rightarrow$
  - $P(\text{test trigram is unseen}) \approx \frac{53}{365} \approx 14.5\%$



## Whither Smoothing?

- Does smoothing matter?
  - No smoothing (MLE estimate): performance will be very poor (zero probabilities will kill you).
  - Difference between OK (bucketed linear interpolation) and the best (modified Kneser-Ney).
    - $\sim 1\%$  absolute in WER for unconstrained speech.
  - No downside to better smoothing (except implementation effort).



## Whither Smoothing?

- Big or small models? e.g., 3-gram or 5-gram?
  - With smaller models, less sparse data issues — better probability estimates?
  - Empirically: bigger is better.
    - With best smoothing, little or no performance degradation if make model too large.
    - With lots of data (100MW+), significant gain from 3-gram to 5-gram.
    - Limiting resource: disk/memory.
  - *Count cutoffs* or *entropy-based pruning* can be used to reduce size of LM.



## Where Are We?

- 6 Grammatical LM's
- 7 N-Gram Models
- 8 Smoothing
- 9 Evaluating Language Models



## Evaluating Language Models

- Best way: plug into ASR system, see how affects WER.
  - Expensive to compute.
- Is there something cheaper that predicts WER well?
  - *Perplexity* (PP) of test data (only needs text).
  - Doesn't really predict WER super well, but has theoretical significance.



## Perplexity

- Compute (geometric) average probability  $p_{\text{avg}} \dots$ 
  - Assigned to each word in test data  $w_1 \dots w_l$  by model  $P(\cdot)$ .
  - $p_{\text{avg}} = \left[ \prod_{i=1}^l P(w_i | w_1 \dots w_{i-1}) \right]^{\frac{1}{l}}$
- 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  $P(\cdot)$ .
  - Related to sampled estimate of Kullback-Leibler (KL) distance between  $P(\cdot)$  and ideal language model.



## Perplexity

- Estimate of human performance (Shannon, 1951)
  - Shannon game — humans guess next letter in text.
  - PP=142 (1.3 bits/letter), uncased, unpunctuated, open vocab.
- Estimate of trigram language model (Brown *et al.* 1992).
  - PP=790 (1.75 bits/letter), cased, punctuated, open vocab.
- ASR systems (uncased, unpunctuated, closed vocab).
  - ~100 for complex domains (Switchboard, Broadcast News).
  - Can be much lower for constrained domains.
- PP depends a lot on vocabulary size.
  - Map all OOV words to single token, the *unknown word*.
  - Larger the vocabulary, larger PP will be.



## Using *N*-Gram Models in ASR

- How to build the HMM used in decoding?
- Actually, an *n*-gram model can be expressed as an FSA.
  - Just pretend it's a big grammar; do as before.
  - Need lots of memory, though.
- Whole subfield of ASR devoted to this task: *search*.
- To be continued ...



## The Road Ahead

- The road behind: small vocabulary ASR.
- Lecture 6-8: large vocabulary ASR.



## 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)?

