Outline for Today

- HMM-GMM acoustic models
  - Mixtures of Gaussians
  - Embedded training. Where a state is progressively:
    - CI Subphone (3ish per phone)
    - Context Dependent (CD) phone (=triphones)
    - State-tying of CD phone

- MFCC feature extraction
- Training data augmentation
Gaussians for Acoustic Modeling

- $P(o|q)$: A Gaussian parameterized by mean and variance:

  - $P(o|q)$ is highest here at mean
  - $P(o|q)$ low here, far from mean

Different means
Using a (univariate) Gaussian as an acoustic likelihood estimator

- Let’s suppose our observation was a single real-valued feature (instead of 39D vector)
- Then if we had learned a Gaussian over the distribution of values of this feature
- We could compute the likelihood of any given observation $o_t$ as follows:

$$b_j(o_t) = \frac{1}{\sqrt{2\pi}\sigma^2_j} \exp \left( -\frac{(o_t - \mu_j)^2}{2\sigma^2_j} \right)$$
Training a Univariate Gaussian

- A (single) Gaussian is characterized by a mean and a variance
- Imagine that we had some training data in which each state was labeled
- We could just compute the mean and variance from the data:

\[ \mu_i = \frac{1}{T} \sum_{t=1}^{T} o_t \text{ s.t. } o_t \text{ is state } i \]

\[ \sigma_i^2 = \frac{1}{T} \sum_{t=1}^{T} (o_t - \mu_i)^2 \text{ s.t. } q_t \text{ is state } i \]
Training Univariate Gaussians

- But we don’t know which observation was produced by which state!
- What we want: to assign each observation vector $o_t$ to every possible state $i$, prorated by the probability the HMM was in state $i$ at time $t$.
- The probability of being in state $i$ at time $t$ is $\gamma_t(i)$!!

$$\bar{\mu}_i = \frac{\sum_{t=1}^{T} \gamma_t(i) o_t}{\sum_{t=1}^{T} \gamma_t(i)}$$

$$\bar{\sigma}_i^2 = \frac{\sum_{t=1}^{T} \gamma_t(i) (o_t - \mu_i)^2}{\sum_{t=1}^{T} \gamma_t(i)}$$
Multivariate Gaussians

- Instead of a single mean $\mu$ and variance $\sigma$:

$$f(x \mid \mu, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)$$

- Vector of observations $x$ modeled by vector of means $\mu$ and covariance matrix $\Sigma$

$$f(x \mid \mu, \Sigma) = \frac{1}{(2\pi)^{D/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu)\right)$$
Multivariate Gaussians

- Defining $\mu$ and $\Sigma$

$$\mu = E(x)$$

$$\Sigma = E[(x - \mu)(x - \mu)^T]$$

- So the $i$-th element of $\Sigma$ is:

$$\sigma_{ij}^2 = E[(x_i - \mu_i)(x_j - \mu_j)]$$
Gaussian Intuitions: Size of $\Sigma$

- $\mu = [0 \ 0]$  \quad $\mu = [0 \ 0]$  \quad $\mu = [0 \ 0]$
- $\Sigma = I$  \quad $\Sigma = 0.6I$  \quad $\Sigma = 2I$
- As $\Sigma$ becomes larger, Gaussian becomes more spread out; as $\Sigma$ becomes smaller, Gaussian more compressed

From Andrew Ng’s lecture notes for CS229
Gaussian Intuitions: Off-diagonal

\[ \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix} \]

Text and figures from Andrew Ng’s lecture notes for CS229
But: assume diagonal covariance

- I.e., assume that the features in the feature vector are uncorrelated
- This isn’t true for FFT features, but is true for MFCC features, as we will see later.
- Computation and storage much cheaper if diagonal covariance.
  - I.e. only diagonal entries are non-zero
  - Diagonal contains the variance of each dimension $\sigma_{ii}^2$
  - So this means we consider the variance of each acoustic feature (dimension) separately
But we’re not there yet

- Single Gaussian may do a bad job of modeling distribution in any dimension:

- Solution: Mixtures of Gaussians

Figure from Chen, Picheney et al slides
Mixture of Gaussians to model a function
GMMs

Summary: each state has a likelihood function parameterized by:

- M Mixture weights
- M Mean Vectors of dimensionality D
- Either
  - M Covariance Matrices of DxD
- Or more likely
  - M Diagonal Covariance Matrices of DxD
  - which is equivalent to
  - M Variance Vectors of dimensionality D
Embedded training

- Train each phone HMM embedded in an entire sentence
- Do word/phone segmentation and alignment automatically as part of training process
HMM-GMM Embedded Training

Transcription: Nine four oh two two

Wavefile

Lexicon:
- one
- two
- three
- eight
- nine
- zero
- oh

n ay n f a o r ow t uw t uw

Raw HMM:

Feature Vectors
Initialization: “Flat start”

• Transition probabilities:
  • set to zero any that you want to be “structurally zero”
  • Set the rest to identical values

• Likelihoods:
  • initialize $\mu$ and $\sigma$ of each state to global mean and variance of all training data
Embedded Training

- Given: phoneset, lexicon, transcribed wavefiles
  - Build a whole sentence HMM for each sentence
  - Initialize A probs to 0.5, or to zero
  - Initialize B probs to global mean and variance
  - Run multiple iterations of Baum Welch
    - During each iteration, we compute forward and backward probabilities
  - Use them to re-estimate A and B
  - Run Baum-Welch until convergence.
Viterbi training

• Baum-Welch training says:
  • We need to know what state we were in, to accumulate counts of a given output symbol $o_t$
  • We’ll compute $\gamma_i(t)$, the probability of being in state $i$ at time $t$, by using forward-backward to sum over all possible paths that might have been in state $i$ and output $o_t$.

• Viterbi training says:
  • Instead of summing over all possible paths, just take the single most likely path
  • Use the Viterbi algorithm to compute this “Viterbi” path
  • Via “forced alignment”
Forced Alignment

- Computing the “Viterbi path” over the training data is called “forced alignment”
- Because we know which word string to assign to each observation sequence.
- We just don’t know the state sequence.
- So we use $a_{ij}$ to constrain the path to go through the correct words
- And otherwise do normal Viterbi
- Result: state sequence!
Phonetic context: different “eh”s

w eh d y eh l b eh n
Modeling phonetic context

- The strongest factor affecting phonetic variability is the neighboring phone
- How to model that in HMMs?
- Idea: have phone models which are specific to context.
- Instead of Context-Independent (CI) phones
- We’ll have Context-Dependent (CD) phones
CD phones: triphones

- Triphones
- Each triphone captures facts about preceding and following phone
- Monophone:
  - p, t, k
- Triphone:
  - iy-p+aa
  - a-b+c means “phone b, preceding by phone a, followed by phone c”
“Need” with triphone models
Word-Boundary Modeling

- Word-Internal Context-Dependent Models
  ‘OUR LIST’:
  SIL AA+R AA-R L+IH L-IH+S IH-S+T S-T
- Cross-Word Context-Dependent Models
  ‘OUR LIST’:
  SIL-AA+R AA-R+L R-L+IH L-IH+S IH-S+T S-T+SIL
- Dealing with cross-words makes decoding harder!
Implications of Cross-Word Triphones

- Possible triphones: $50 \times 50 \times 50 = 125,000$
- How many triphone types actually occur?
- 20K word WSJ Task, numbers from Young et al
- Cross-word models: need 55,000 triphones
- But in training data only 18,500 triphones occur!
- Need to generalize models
Modeling phonetic context: some contexts look similar

w iy  r iy  m iy  n iy
Solution: State Tying

- Young, Odell, Woodland 1994
- Decision-Tree based clustering of triphone states
- States which are clustered together will share their Gaussians
- We call this “state tying”, since these states are “tied together” to the same Gaussian.
Triphone decision tree clustering

Phone /ih/ beg. state

Left nasal?

Yes

Left fricative?

No

Right liquid?

Right /l/?

Cluster A: n-ih+l₀
ng-ih+l₀
m-ih+l₀

A

Yes

Cluster B: n-ih+r₀
ng-ih+r₀
m-ih+r₀
n-ih+w₀

B

No

C

D

No
Summary: Acoustic Modeling for LVCSR

- Increasingly sophisticated models
- For each state:
  - Gaussians
  - Multivariate Gaussians
  - Mixtures of Multivariate Gaussians
- Where a state is progressively:
  - CI Phone
  - CI Subphone (3ish per phone)
  - CD phone (=triphones)
  - State-tying of CD phone
- Viterbi training
- Neural network acoustic models after all of the above
Summary: ASR Architecture

- Five easy pieces: ASR Noisy Channel architecture
  - Feature Extraction:
    - 39 “MFCC” features
  - Acoustic Model:
    - Gaussians for computing $p(o|q)$
  - Lexicon/Pronunciation Model
    - HMM: what phones can follow each other
  - Language Model
    - N-grams for computing $p(w_i|w_{i-1})$
  - Decoder
    - Viterbi algorithm: dynamic programming for combining all these to get word sequence from speech!
MFCC

- Mel-Frequency Cepstral Coefficient (MFCC)
- Most widely used spectral representation in ASR
Mel-scale

- Human hearing is not equally sensitive to all frequency bands
- Less sensitive at higher frequencies, roughly > 1000 Hz
- I.e. human perception of frequency is non-linear:
Mel Filter Bank Processing

- Mel Filter bank
  - Roughly uniformly spaced before 1 kHz
  - Logarithmic scale after 1 kHz
Delta and double-delta

- Derivative: in order to obtain temporal information

\[ d(t) = \frac{c(t+1) - c(t-1)}{2} \]
Typical MFCC features

- Window size: 25ms
- Window shift: 10ms
- Pre-emphasis coefficient: 0.97
- MFCC:
  - 12 MFCC (mel frequency cepstral coefficients)
  - 1 energy feature
  - 12 delta MFCC features
  - 12 double-delta MFCC features
  - 1 delta energy feature
  - 1 double-delta energy feature
- Total 39-dimensional features
Why is MFCC so popular?

- Efficient to compute
- Incorporates a perceptual Mel frequency scale
- Separates the source and filter
- IDFT(DCT) de-correlates the features
  - Necessary for diagonal assumption in GMM modeling
- There are alternatives like PLP
- Choice matters less for neural network acoustic models
Acoustic Modeling with GMMs

Transcription: Samson
Sub-phones: 942 – 6 – 37 – 8006 – 4422 …

Hidden Markov Model (HMM):

Acoustic Model:

Audio Input:

GMM models: P(x|s)
  x: input features
  s: HMM state
DNN Hybrid Acoustic Models

Transcription: Samson
Sub-phones: 942 – 6 – 37 – 8006 – 4422 …

Hidden Markov Model (HMM):

Acoustic Model:

Audio Input:

Use a DNN to approximate: P(s|x)
Apply Bayes’ Rule: P(x|s) = P(s|x) * P(x) / P(s)
DNN * Constant / State prior
Training Augmentation: SpecAugment

1. Time warping (image warp)
2. Frequency masking
3. Time masking

Mix using different policies

Implementations available for PyTorch

(Park et al, ICASSP 2020)
## Training Augmentation: SpecAugment

(Park et al, ICASSP 2020)

Table 5: *Switchboard 300h WERs* (%).

<table>
<thead>
<tr>
<th>Method</th>
<th>No LM</th>
<th>With LM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SWBD</td>
<td>CH</td>
</tr>
<tr>
<td><strong>HMM</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Veselý et al., (2013) [41]</td>
<td>12.9</td>
<td>24.5</td>
</tr>
<tr>
<td>Hadian et al., (2018) [42]</td>
<td>9.3</td>
<td>18.9</td>
</tr>
<tr>
<td><strong>CTC</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zweig et al., (2017) [43]</td>
<td>24.7</td>
<td>37.1</td>
</tr>
<tr>
<td>Audhkhasi et al., (2018) [44]</td>
<td>20.8</td>
<td>30.4</td>
</tr>
<tr>
<td>Audhkhasi et al., (2018) [45]</td>
<td>14.6</td>
<td>23.6</td>
</tr>
<tr>
<td><strong>LAS</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lu et al., (2016) [46]</td>
<td>26.8</td>
<td>48.2</td>
</tr>
<tr>
<td>Toshniwal et al., (2017) [47]</td>
<td>23.1</td>
<td>40.8</td>
</tr>
<tr>
<td>Zeyer et al., (2018) [38]</td>
<td>11.9</td>
<td>23.7</td>
</tr>
<tr>
<td><strong>Our Work</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAS</td>
<td>11.2</td>
<td>21.6</td>
</tr>
<tr>
<td>LAS + SpecAugment (SM)</td>
<td><strong>7.2</strong></td>
<td>14.6</td>
</tr>
<tr>
<td>LAS + SpecAugment (SS)</td>
<td><strong>7.3</strong></td>
<td><strong>14.4</strong></td>
</tr>
</tbody>
</table>
Acoustic Model Adaptation

- Shift the means and variances of Gaussians to better match the input feature distribution
  - Maximum Likelihood Linear Regression (MLLR)
  - Maximum A Posteriori (MAP) Adaptation
- For both speaker adaptation and environment adaptation
- Widely used!
Maximum Likelihood Linear Regression (MLLR)


- Given:
  - a trained AM
  - a small “adaptation” dataset from a new speaker

- Learn new values for the Gaussian mean vectors
  - Not by just training on the new data (too small)
  - But by learning a linear transform which moves the means.
Maximum Likelihood Linear Regression (MLLR)

- Estimates a linear transform matrix ($W$) and bias vector ($\omega$) to transform HMM model means:

$$\mu_{\text{new}} = W_r \mu_{\text{old}} + \omega_r$$

- Transform estimated to maximize the likelihood of the adaptation data
- New equation for output likelihood

\[ b_j(o_t) = \frac{1}{(2\pi)^{n/2} |\Sigma_j|^{1/2}} \exp\left( -\frac{1}{2} (o_t - (W\mu_j + \omega)) |\Sigma_j|^{-1} (o_t - (W\mu_j + \omega))^T \right) \]
Q: Why is estimating a linear transform from adaptation data different than just training on the data?

A: Even from a very small amount of data we can learn 1 single transform for all triphones! So small number of parameters.

A2: If we have enough data, we could learn more transforms (but still less than the number of triphones). One per phone (~50) is often done.
MLLR: Learning

- Given
  - a small labeled adaptation set (a couple sentences)
  - a trained AM
- Do forward-backward alignment on adaptation set to compute state occupation probabilities $\gamma_j(t)$.
- $W$ can now be computed by solving a system of simultaneous equations involving $\gamma_j(t)$.
MLLR performance on baby task (RM) (Leggetter and Woodland 1995)

Figure 2. Full matrix maximum likelihood linear regression using global regression class. (-----), Speaker independent; (---------), speaker dependent; (----), speaker adapted.
MLLR doesn’t need supervised adaptation set!

Figure 3. Supervised vs. unsupervised adaptation using maximum likelihood linear regression. (⋯⋯), Speaker independent; (———), speaker dependent; (——), supervised adapted; (---), unsupervised adapted.
Maximum A Posteriori Adaptation (MAP)

- MAP Adaptation can only be applied Gaussians that are “seen” in the test data,

\[
\mu_{\text{new}} = \frac{\hat{N}}{\hat{N} + \alpha} \hat{m}_{\text{obs}} + \frac{\alpha}{\hat{N} + \alpha} \mu_{\text{old}}
\]

- $\hat{N}$: Number of frames of adaptation data
- $\alpha$: Weight for prior estimate of old mean
- $\hat{m}_{\text{obs}}$: Mean vector of adaptation data assigned to Gauss.
Performance of MLLR and MAP

Speaker independent

Speaker dependent

Error Rate

Number of Adaptation Utterances

MAP

MLLR

MLLR+MAP

Slide from Bryan Pellom after Huang et al
Summary

- MLLR: works on small amounts of adaptation data
- MAP: Maximum A Posterior Adaptation
  - Works well on large adaptation sets
- Acoustic adaptation techniques are quite successful at dealing with speaker variability
- If we can get 10 seconds with the speaker.
Young et al. state tying
State tying/clustering

- How do we decide which triphones to cluster together?
- Use phonetic features (or ‘broad phonetic classes’)
  - Stop
  - Nasal
  - Fricative
  - Sibilant
  - Vowel
  - Lateral
## Decision tree for clustering triphones for tying

<table>
<thead>
<tr>
<th>Feature</th>
<th>Phones</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop</td>
<td>b d g k p t</td>
</tr>
<tr>
<td>Nasal</td>
<td>m n ng</td>
</tr>
<tr>
<td>Fricative</td>
<td>ch dh f jh s sh th v z zh</td>
</tr>
<tr>
<td>Liquid</td>
<td>l r w y</td>
</tr>
<tr>
<td>Vowel</td>
<td>aa ae ah ao aw ax axr ay eh er ey ih ix iy ow oy uh uw</td>
</tr>
<tr>
<td>Front Vowel</td>
<td>ae eh ih ix iy</td>
</tr>
<tr>
<td>Central Vowel</td>
<td>aa ah ao axr er</td>
</tr>
<tr>
<td>Back Vowel</td>
<td>ax ow uh uw</td>
</tr>
<tr>
<td>High Vowel</td>
<td>ih ix iy uh uw</td>
</tr>
<tr>
<td>Rounded</td>
<td>ao ow oy uh uw w</td>
</tr>
<tr>
<td>Reduced</td>
<td>ax axr ix</td>
</tr>
<tr>
<td>Unvoiced</td>
<td>ch f hh k p s sh t th</td>
</tr>
<tr>
<td>Coronal</td>
<td>ch d dh jh l n r s sh t th z zh</td>
</tr>
</tbody>
</table>
State Tying: Young, Odell, Woodland 1994

(1) Train monophone single Gaussian models

(2) Clone monophones to triphones

(3) Cluster and tie triphones

(4) Expand to GMMs
Viterbi training equations

Viterbi

\[
\hat{a}_{ij} = \frac{n_{ij}}{n_i}
\]

For all pairs of emitting states, 1 \leq i, j \leq N

\[
\hat{b}_j(v_k) = \frac{n_j(s.t. o_t = v_k)}{n_j}
\]

Baum-Welch

\[
\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \gamma_t(i, j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \gamma_t(i, j)}
\]

\[
\hat{b}_j(v_k) = \frac{\sum_{t=1}^{T} s.t. o_t = v_k \xi_j(t) \gamma}{\sum_{t=1}^{T} \xi_j(t) \gamma}
\]

Where \( n_{ij} \) is number of frames with transition from \( i \) to \( j \) in best path
And \( n_j \) is number of frames where state \( j \) is occupied
Discrete Representation of Signal

- Represent continuous signal into discrete form.

Image from Bryan Pellom
Sampling

If measure at green dots, will see a lower frequency wave and miss the correct higher frequency one!

Original signal in red:
WAV format

Many formats, trade-offs in compression, quality

Nice sound manipulation tool: Sox
http://sox.sourceforge.net/
convert speech formats
Windowing

A \sim 20 - 25 \text{ ms}

B \sim 10 \text{ ms}

Image from Bryan Pellom
MFCC

\[ y_t = \left\{ y_t(j), e_t, \Delta y_t(j), \Delta e_t, \Delta^2 y_t(j), \Delta^2 e_t \right\} \]

Speech signal \( x(n) \) → Pre-emphasis → \( x'(n) \) → \( x_t(n) \) → DFT → \( X_t(k) \) → Mel filter-bank → \( Y_t(m) \)

\( Y_t'(m) \) → IDFT → MFCC
Discrete Fourier Transform computing a spectrum

- A 25 ms Hamming-windowed signal from [iy]
- And its spectrum as computed by DFT (plus other smoothing)
Mel-filter Bank Processing

- Apply the bank of Mel-scaled filters to the spectrum
- Each filter output is the sum of its filtered spectral components

\[ x_t(n) \rightarrow \text{DFT} \rightarrow X_t(k) \]

\[ n = 0,1,\ldots,L-1 \quad k = 0,1,\ldots,\frac{L}{2} - 1 \]
MFCC

Speech signal $x(n)$ → Pre-emphasis $x'(n)$ → DFT $X_t(k)$ → Mel filter-bank

$y_t = \left\{ y_t(j), e_t, \Delta y_t(j), \Delta e_t, \Delta^2 y_t(j), \Delta^2 e_t \right\}$ → derivatives $e_t$ → Log($||^2$) $Y_t(m)$ → IDFT $Y_t'(m)$
Log energy computation

- Compute the logarithm of the square magnitude of the output of Mel-filter bank
The Cepstrum

- One way to think about this
  - Separating the source and filter
  - Speech waveform is created by
    - A glottal source waveform
    - Passes through a vocal tract which because of its shape has a particular filtering characteristic

- Remember articulatory facts from lecture 2:
  - The vocal cord vibrations create harmonics
  - The mouth is an amplifier
  - Depending on shape of oral cavity, some harmonics are amplified more than others
We care about the filter not the source

- Most characteristics of the source
  - F0
  - Details of glottal pulse
- Don’t matter for phone detection
- What we care about is the filter
  - The exact position of the articulators in the oral tract
- So we want a way to separate these
  - And use only the filter function
The Cepstrum

- The spectrum of the log of the spectrum
Another advantage of the Cepstrum

- DCT produces highly uncorrelated features

- If we use only the diagonal covariance matrix for our Gaussian mixture models, we can only handle uncorrelated features.

- In general we’ll just use the first 12 cepstral coefficients (we don’t want the later ones which have e.g. the F0 spike)