Announcements, Assignments, and Reminders

- Visit the URL links that were covered in previous lectures.
Cumulative Outstanding Reading

- Read chapters 1 and 2 in our book (Huang, Acero, Hon, “Spoken Language Processing”).
- Read chapters 3 and 4 in our book (Huang, Acero, Hon, “Spoken Language Processing”).
- Read Chapter 6 in our book (Huang, Acero, Hon, “Spoken Language Processing”).
- Read HMM sections in our book (Huang, Acero, Hon, “Spoken Language Processing”).
- Read Chapter 9 in our book (Huang, Acero, Hon, “Spoken Language Processing”).
- Read Chapter 11 in our book (Huang, Acero, Hon, “Spoken Language Processing”).
On Final Project

- Will be held Monday, June 10th, 2013
- time/place: TBD
- Project should ideally be on some aspect of the material we have learnt, some aspect of speech processing or recognition. Possible good projects include:
  - A modern advanced paper summary, of papers that we are not going to cover in this class.
  - A new idea of your own, new algorithms and/or theoretical results.
  - Implement a speech recognition system in HTK or some other system.
  - new speech coding, speech application, or
  - application of ideas from speech recognition to other types of data (but must explain in speech terminology).
On Final Project

- The ideal project should be research-oriented
- Ideal project would lead to a conference and/or journal paper.
- Fine to combine it with your own research.
- Deadline every Monday, 5:00pm up until day of final project 6/10.
Final Project - Toolkits

- HTK - HMM toolkit (Cambridge, UK)
- CMU-Sphinx http://cmusphinx.sourceforge.net/ - HMM-based Speech recognition toolkit, CMU
- GMTK - general DBN toolkit, originally for speech but useful in general.
- Matlab - good for small problems and for speech processing ideas (but doesn’t scale to larger systems and/or data).
Final Project - pending deadlines

Every Monday from now up until June 10th (our final presentations day). All should be submitted to our dropbox (https://catalyst.uw.edu/collectit/dropbox/bilmes/26924)

Specific deadlines are as follows:

- May 27th: 11:45pm: project proposal update (1 page max).
- June 3rd: 11:45pm: project status update (1 page max).
- June 10th: 11:00am: final project report (4 pages max).

Note, all deadlines are at 11:45pm at night except for last one which is at 11:00am in the morning.

Office hours and/or email if you have any questions.
Review

- HMM gradients
- HMM conditional likelihood based training.
- Extended Baum Welch
Outline of today

- Pronunciation modeling.
Good books (for today)

- our book (Huang, Acero, Hon, “Spoken Language Processing”)  
- Deller et. al. “Discrete-time Processing of speech signals”  
- O'Shaughnessy, “Speech Communications”  
- J. Bilmes, “What HMMs can do”, 2010
Extended Baum-Welch

Extended Baum-Welch

- Extended Baum Welch: maximizes rational functions of polynomials of probabilities with non-negative coefficients (like an MMIE criterion).
Extended Baum-Welch

- Extended Baum Welch: maximizes rational functions of polynomials of probabilities with non-negative coefficients (like an MMIE criterion).

rational functions of polynomials of probabilities with non-negative coefficients, examples:

\[ R(p_1, p_2, p_3) = \frac{3p_1p_2}{p_1^2 + p_2^2 + 2p_3^2} \]  \hspace{1cm} (15.1)

where \( 0 \leq p_i \leq 1 \) for all \( i \) and \( \sum_i p_i = 1 \).
Extended Baum-Welch

- General rational functions of polynomials of probabilities with non-neg coefficients, examples:

\[ P(\Lambda) = P(\{\Lambda_{ij}\}), i = 1 \ldots p; j = 1 \ldots q_i \]  

(15.2)
General rational functions of polynomials of probabilities with non-neg coefficients, examples:

\[ P(\Lambda) = P(\{\Lambda_{ij}\}), \quad i = 1 \ldots p; \quad j = 1 \ldots q_i \quad (15.2) \]

Polynomials defined over multiple simplicies, \( \triangle \):

\[ \Lambda \in \triangle = \left\{ \lambda_{ij} : \lambda_{ij} \geq 0, \sum_{j=1}^{q_i} \lambda_{ij} = 1 \right\} \quad (15.3) \]
Extended Baum-Welch

- General rational functions of polynomials of probabilities with non-neg coefficients, examples:

\[ P(\Lambda) = P(\{\Lambda_{ij}\}), i = 1 \ldots p; j = 1 \ldots q_i \]  

(15.2)

- Polynomials defined over multiple simplicies, \( \triangle \):

\[ \Lambda \in \triangle = \left\{ \lambda_{ij} : \lambda_{ij} \geq 0, \sum_{j=1}^{q_i} \lambda_{ij} = 1 \right\} \]  

(15.3)

- We have ratios of such polynomials \( S_1 \) and \( S_2 \):

\[ R(\Lambda) = S_1(\Lambda)/S_2(\Lambda) \]  

(15.4)
Extended Baum-Welch

How is it that our problem is in this form?

\[ 2I_\lambda(q;x) = \frac{p_\lambda(x|q)}{p_\lambda(x)} = \frac{p_\lambda(x|q)}{\sum_{q'} p(q')p_\lambda(x|q')} \]  

(15.5)

and for an HMM

\[ p_\lambda(x_{1:T}|q) = \sum_{q_1:T} \prod_{t} p_\lambda(x_t|q_t)p_\lambda(q_t|q_{t-1}) \]  

(15.6)
Extended Baum-Welch

- How is it that our problem is in this form?

\[ 2I_\lambda(q;x) = \frac{p_\lambda(x|q)}{p_\lambda(x)} = \frac{p_\lambda(x|q)}{\sum_{q'} p(q')p_\lambda(x|q')} \quad (15.5) \]

and for an HMM

\[ p_\lambda(x_{1:T}|q) = \sum_{q_{1:T}} \prod_{t} p_\lambda(x_t|q_t)p_\lambda(q_t|q_{t-1}) \quad (15.6) \]

- Therefore, for discrete observation parameters, both numerator and denominator are of the right form (polynomials with non-negative coefficients with positive coefficients)
Extended Baum-Welch

How is it that our problem is in this form?

\[ 2I_\lambda(q;x) = \frac{p_\lambda(x|q)}{p_\lambda(x)} = \frac{p_\lambda(x|q)}{\sum_{q'} p(q') p_\lambda(x|q')} \quad (15.5) \]

and for an HMM

\[ p_\lambda(x_1:T|q) = \sum_{q_1:T} \prod_{t} p_\lambda(x_t|q_t) p_\lambda(q_t|q_{t-1}) \quad (15.6) \]

Therefore, for discrete observation parameters, both numerator and denominator are of the right form (polynomials with non-negative coefficients with positive coefficients)

For Gaussian densities, this has been extended (see Normandin'91), we don't cover this today.
Growth transformation

Definition: Growth transformations $T$ of $\triangle$ for $R(\Lambda)$: $\forall \lambda \in \triangle$, let $\xi \leftarrow T(\lambda)$. Then we’ll have $R(\xi) > R(\lambda)$ whenever $\lambda \neq \xi$. 

$T(\cdot)$ will either increase $R(\cdot)$ or leave it alone.

Our goal: find a growth transformation for $R(\cdot)$ show that for a particular polynomial, $P$, a growth transformation for $P$ is also a growth transformation for $R$ transform the polynomial so that it is applicable to a theorem that gives us a growth transformation for a suitable polynomial apply it to MMIE estimation with discrete parameters.
Growth transformation

- Definition: Growth transformations $T$ of $\triangle$ for $R(\Lambda)$: $\forall \lambda \in \triangle$, let $\xi \leftarrow T(\lambda)$. Then we’ll have $R(\xi) > R(\lambda)$ whenever $\lambda \neq \xi$.
- $T(\cdot)$ will either increase $R(\cdot)$ or leave it alone.
Definition: Growth transformations $T$ of $\triangle$ for $R(\Lambda)$: $\forall \lambda \in \triangle$, let $\xi \leftarrow T(\lambda)$. Then we’ll have $R(\xi) > R(\lambda)$ whenever $\lambda \neq \xi$.

$T(\cdot)$ will either increase $R(\cdot)$ or leave it alone.

Our goal:
- find a growth transformation for $R(\cdot)$
- show that for a particular polynomial, $P$, a growth transformation for $P$ is also a growth transformation for $R$
- transform the polynomial so that it is applicable to a theorem that gives us a growth transformation for a suitable polynomial
- apply it to MMIE estimation with discrete parameters
A **homogeneous** polynomial means that degree of each monomial is the same in the sum. Ex: $x^3 + x^2y + xy^2 + y^3$ but not $x^3 + x^2y + xy^2 + y^2$

**Theorem 15.3.1 (Baum67)**

Let $P(\Lambda)$ be a homogeneous polynomial with non-negative coefficients, degree $d$, defined on $\triangle$ such that

$$\sum_{j=1}^{q_i} \lambda_{ij} \frac{\partial P}{\partial \Lambda_{ij}}(\lambda_{ij}) \neq 0, \forall i$$  \hspace{1cm} (15.7)

Define transformation $\xi = T(\lambda)$ where $ij$ mapping is defined as:

$$\xi_{ij} = \frac{\lambda_{ij} \frac{\partial P}{\partial \Lambda_{ij}}(\lambda_{ij})}{\sum_{j=1}^{q_i} \lambda_{ij} \frac{\partial P}{\partial \Lambda_{ij}}(\lambda_{ij})}$$  \hspace{1cm} (15.8)

Then $P(T(\lambda)) > P(\lambda)$ unless $T(\lambda) = \lambda$. 

---

**Prof. Jeff Bilmes**


L15 F15/54 (pg.23/184)
This gives us a growth transformation, but for the wrong type of object.
Extended Baum-Welch

- This gives us a growth transformation, but for the wrong type of object.
- Theorem applies to homogeneous polys of degree $d$ with non-negative coefficients.
This gives us a growth transformation, but for the wrong type of object.

Theorem applies to homogeneous polys of degree $d$ with non-negative coefficients.

We’ve got: 1) $R()$, a ratio of polynomials; 2) We might have negative coefficients (as we will see); 3) The polynomials might be non-homogeneous.
Extended Baum-Welch

- This gives us a growth transformation, but for the wrong type of object.
- Theorem applies to homogeneous polys of degree $d$ with non-negative coefficients.
- We’ve got: 1) $R()$, a ratio of polynomials; 2) We might have negative coefficients (as we will see); 3) The polynomials might be non-homogeneous
- We define a 3-step procedure to go from $R()$ to a $P()$ that satisfies the theorem, but also that if $T()$ is a growth transformation for $P()$, it is also one for $R()$
Extended Baum-Welch

Step 1: dealing with ratios.

- Move away from ratio of polynomials to just polynomials, where \( R(\lambda) = S_1(\lambda)/S_2(\lambda) \), using the following:

\[
P_\lambda(\Lambda) \triangleq S_1(\Lambda) - R(\lambda)S_2(\Lambda)
\]  

(15.9)
Extended Baum-Welch

Step 1: dealing with ratios.

- Move away from ratio of polynomials to just polynomials, where \( R(\lambda) = S_1(\lambda)/S_2(\lambda) \), using the following:

\[
P_\lambda(\Lambda) \triangleq S_1(\Lambda) - R(\lambda)S_2(\Lambda)
\]  \quad (15.9)

- Therefore, if \( P_\lambda(\xi) > P_\lambda(\lambda) = 0 \), then \( R(\xi) > R(\lambda) \), seen by solving for \( R(\xi) = S_1(\xi)/S_2(\xi) \) in:

\[
P_\lambda(\xi) = S_1(\xi) - R(\lambda)S_2(\xi) > 0
\]  \quad (15.10)
Extended Baum-Welch

Step 1: dealing with ratios.

- Move away from ratio of polynomials to just polynomials, where
  \( R(\lambda) = S_1(\lambda)/S_2(\lambda) \), using the following:

  \[
  P_\lambda(\Lambda) \triangleq S_1(\Lambda) - R(\lambda)S_2(\Lambda) \tag{15.9}
  \]

- Therefore, if \( P_\lambda(\xi) > P_\lambda(\lambda) = 0 \), then \( R(\xi) > R(\lambda) \), seen by solving for \( R(\xi) = S_1(\xi)/S_2(\xi) \) in:

  \[
  P_\lambda(\xi) = S_1(\xi) - R(\lambda)S_2(\xi) > 0 \tag{15.10}
  \]

- Given growth transform \( T_\lambda(\cdot) \) for polynomial \( P_\lambda(\xi) \), so that
  \( P_\lambda(T_\lambda(\xi)) > P_\lambda(\xi) \) (unless \( T_\lambda(\xi) = \xi \)). Then define \( T(\lambda) = T_\lambda(\lambda) \).
Extended Baum-Welch

Step 1: dealing with ratios.

- Move away from ratio of polynomials to just polynomials, where 
  \[ R(\lambda) = \frac{S_1(\lambda)}{S_2(\lambda)} \], using the following:

  \[
P_\lambda(\Lambda) \triangleq S_1(\Lambda) - R(\lambda)S_2(\Lambda) \quad (15.9)
  \]

- Therefore, if \( P_\lambda(\xi) > P_\lambda(\lambda) = 0 \), then \( R(\xi) > R(\lambda) \), seen by solving for \( R(\xi) = \frac{S_1(\xi)}{S_2(\xi)} \) in:

  \[
P_\lambda(\xi) = S_1(\xi) - R(\lambda)S_2(\xi) > 0 \quad (15.10)
  \]

- Given growth transform \( T_\lambda(\cdot) \) for polynomial \( P_\lambda(\xi) \), so that 
  \( P_\lambda(T_\lambda(\xi)) > P_\lambda(\xi) \) (unless \( T_\lambda(\xi) = \xi \)). Then define \( T(\lambda) = T_\lambda(\lambda) \).

- Then, we have that if \( P_\lambda(T_\lambda(\lambda)) > P_\lambda(\lambda) \), then \( R(T(\lambda)) > R(\lambda) \).
Extended Baum-Welch

Step 1: dealing with ratios.

- Move away from ratio of polynomials to just polynomials, where $R(\lambda) = S_1(\lambda)/S_2(\lambda)$, using the following:

  \[
P_\lambda(\Lambda) \triangleq S_1(\Lambda) - R(\lambda)S_2(\Lambda) \tag{15.9}
  \]

- Therefore, if $P_\lambda(\xi) > P_\lambda(\lambda) = 0$, then $R(\xi) > R(\lambda)$, seen by solving for $R(\xi) = S_1(\xi)/S_2(\xi)$ in:

  \[
P_\lambda(\xi) = S_1(\xi) - R(\lambda)S_2(\xi) > 0 \tag{15.10}
  \]

- Given growth transform $T_\lambda(\cdot)$ for polynomial $P_\lambda(\xi)$, so that $P_\lambda(T_\lambda(\xi)) > P_\lambda(\xi)$ (unless $T_\lambda(\xi) = \xi$). Then define $T(\lambda) = T_\lambda(\lambda)$.

- Then, we have that if $P_\lambda(T_\lambda(\lambda)) > P_\lambda(\lambda)$, then $R(T(\lambda)) > R(\lambda)$.

- So if we’ve got a growth transform for $P(\lambda) \triangleq P_\lambda(\lambda)$, then we’ve got one for $R(\lambda)$.

We still need to deal with non-negative coefficients and homogeneity (negativity can arise due to “-”).
Extended Baum-Welch

Step 1: dealing with ratios.

- Move away from ratio of polynomials to just polynomials, where
  \[ R(\lambda) = S_1(\lambda)/S_2(\lambda), \]
  using the following:
  \[
P_\lambda(\Lambda) \triangleq S_1(\Lambda) - R(\lambda)S_2(\Lambda)
\] (15.9)

- Therefore, if \( P_\lambda(\xi) > P_\lambda(\lambda) = 0 \), then \( R(\xi) > R(\lambda) \), seen by solving for \( R(\xi) = S_1(\xi)/S_2(\xi) \) in:
  \[
P_\lambda(\xi) = S_1(\xi) - R(\lambda)S_2(\xi) > 0
\] (15.10)

- Given growth transform \( T_\lambda(\cdot) \) for polynomial \( P_\lambda(\xi) \), so that \( P_\lambda(T_\lambda(\xi)) > P_\lambda(\xi) \) (unless \( T_\lambda(\xi) = \xi \)). Then define \( T(\lambda) = T_\lambda(\lambda) \).
- Then, we have that if \( P_\lambda(T_\lambda(\lambda)) > P_\lambda(\lambda) \), then \( R(T(\lambda)) > R(\lambda) \).
- So if we’ve got a growth transform for \( P(\lambda) \triangleq P_\lambda(\lambda) \), then we’ve got one for \( R(\lambda) \).
- We still need to deal with non-negative coefficients and homogeneity (negativity can arise due to “-“)
Extended Baum-Welch

Step 2: dealing with negative coefficients.

- We define new polynomial

\[ P'(\Lambda) \triangleq P(\Lambda) + C(\Lambda) \]  \hspace{1cm} (15.11)

where, if \( a \) is \( P \)'s minimal negative coefficient (or \( a = 0 \) if none), and \( d \) is \( P \)'s degree,

\[
C(\Lambda) = -a \left( \sum_{i=1}^{p} \sum_{j=1}^{p_i} \Lambda_{ij} + 1 \right)^d = -a(p + 1)^d \]  \hspace{1cm} (15.12)
Extended Baum-Welch

Step 2: dealing with negative coefficients.

- We define new polynomial

\[ P'(\Lambda) \triangleq P(\Lambda) + C(\Lambda) \]  \hspace{1cm} (15.11)

where, if \( a \) is \( P \)'s minimal negative coefficient (or \( a = 0 \) if none), and \( d \) is \( P \)'s degree,

\[ C(\Lambda) = -a \left( \sum_{i=1}^{p} \sum_{j=1}^{p_i} \Lambda_{ij} + 1 \right)^d = -a(p + 1)^d \]  \hspace{1cm} (15.12)

Therefore, we’ve added a constant to \( P \) to get \( P' \), gotten a non-negative coefficient polynomial as a result, and have not changed the effect of any growth transformations.
Extended Baum-Welch

Step 3: Dealing with non-homogeneous polynomials while simultaneously preserving growth transforms

- We form a new polynomial

\[
P''(\Psi) = \Psi_{p+1,1}^d P'(\{\Psi_{ij}/\Psi_{p+1,1}\}) \tag{15.13}
\]

which is variable substitution with:

\[
\Lambda_{ij} = \Psi_{ij}/\Psi_{p+1,1} \quad \text{and constraint} \quad \Psi_{p+1,1} = 1 \tag{15.14}
\]
Extended Baum-Welch

Step 3: Dealing with non-homogeneous polynomials while simultaneously preserving growth transforms

- We form a new polynomial

\[ P''(\Psi) = \Psi_{d+1,1}^{p+1,1} P'(\{\Psi_{ij}/\Psi_{p+1,1}\}) \]  

(15.13)

which is variable substitution with:

\[ \Lambda_{ij} = \Psi_{ij}/\Psi_{p+1,1} \]  

and constraint \( \Psi_{p+1,1} = 1 \)  

(15.14)

- New set of constrained simplicies:

\[ \Psi \in \Delta' = \left\{ \psi_{ij} : \forall i, j, \psi_{ij} \geq 0, \text{ and } \forall i, \sum_{j=1}^{q_i} \psi_{ij} = 1 \right\} \]  

(15.15)

for \( i = 1, \ldots, p + 1, j = 1, \ldots, q_i, \psi_{p+1,1} = 1 \).
Extended Baum-Welch

Step 3: Dealing with non-homogeneous polynomials while simultaneously preserving growth transforms

- We form a new polynomial

\[ P''(\Psi) = \Psi_{p+1,1}^d P'(\{\Psi_{ij}/\Psi_{p+1,1}\}) \]  (15.13)

which is variable substitution with:

\[ \Lambda_{ij} = \Psi_{ij}/\Psi_{p+1,1} \]  \text{ and constraint } \Psi_{p+1,1} = 1 \]  (15.14)

- New set of constrained simplicies:

\[ \Psi \in \triangle' = \left\{ \psi_{ij} : \forall i, j, \psi_{ij} \geq 0, \text{ and } \forall i, \sum_{j=1}^{q_i} \psi_{ij} = 1 \right\} \]  (15.15)

for \( i = 1, \ldots, p + 1, j = 1, \ldots, q_i, \psi_{p+1,1} = 1 \).

- Note, with \( q_{p+1} = 1 \) means that \( \psi_{p+1,1} = 1 \).
Therefore, $\triangle$ and $\triangle'$ are isomorphic, and there is a 2D bijection between $\lambda$ and $\psi$. 
Therefore, $\triangle$ and $\triangle'$ are isomorphic, and there is a 2D bijection between $\lambda$ and $\psi$.

Any growth function in $\triangle'$ for $P''$ will thus be a growth function in $\triangle$ for $P'$ (and by step 2 a growth function for $P$, and by step 1 a growth function for $R$).
Therefore, $\triangle$ and $\triangle'$ are isomorphic, and there is a 2D bijection between $\lambda$ and $\psi$.

Any growth function in $\triangle'$ for $P''$ will thus be a growth function in $\triangle$ for $P'$ (and by step 2 a growth function for $P$, and by step 1 a growth function for $R$).

But $P''$ satisfies the criterion for Baum’s theorem, so we construct a growth function for $P''$ and use it for $R$ (undoing the steps 1-3 when necessary).
Therefore, $\triangle$ and $\triangle'$ are isomorphic, and there is a 2D bijection between $\lambda$ and $\psi$.

Any growth function in $\triangle'$ for $P''$ will thus be a growth function in $\triangle$ for $P'$ (and by step 2 a growth function for $P$, and by step 1 a growth function for $R$).

But $P''$ satisfies the criterion for Baum’s theorem, so we construct a growth function for $P''$ and use it for $R$ (undoing the steps 1-3 when necessary).

We can combine steps 1-3 and Baum’s theorem into a new theorem that gives us growth functions for rational functions $R()$, as we do next:
Theorem 15.3.2 (Gopalakrishnan91)

Assume $R(\Lambda)$ is a rational function of polynomials in $\Lambda_{ij}$. Then $\exists a_R$ such that for $C \geq a_R$, the following function $T^C(\cdot)$ is a growth transformation in $\triangle$ for $R()$.

$$[T^C(\lambda)]_{ij} = \frac{\lambda_{ij} \left( \frac{\partial P_{\lambda}}{\partial \Lambda_{ij}}(\lambda) + C \right)}{\sum_{j=1}^{q_i} \lambda_{ij} \left( \frac{\partial P_{\lambda}}{\partial \Lambda_{ij}}(\lambda) + C \right)}$$

(15.16)

where $a_R = ad(p + 1)^{d-1}$ and $a = \max_{\lambda} a_{\lambda}$ and where $a_{\lambda}$ is minimal negative coefficient for all polynomials over all $\lambda$. 

Extended Baum-Welch
We can apply this to MMIE training, where in this case we get (for uniform word priors)

\[
Z_\lambda = 2^I_\lambda(q;x) = \frac{p_\lambda(x|q)}{p_\lambda(x)} = \frac{p_\lambda(x|q)}{\sum_{q'} p_\lambda(x|q')} \quad (15.17)
\]
We can apply this to MMIE training, where in this case we get (for uniform word priors)

\[ Z_\lambda = 2^{I_\lambda(q;x)} = \frac{p_\lambda(x|q)}{p_\lambda(x)} = \frac{p_\lambda(x|q)}{\sum_{q'} p_\lambda(x|q')} \quad (15.17) \]

In discrete case we get update equations:

\[ a_{ij}^{t+1} = \frac{a_{ij}^t \left( \frac{\partial \log Z_\lambda}{\partial a_{ij}} (\lambda) + C(\lambda) \right)}{\sum_{j=1}^{q_i} a_{ij}^t \left( \frac{\partial \log Z_\lambda}{\partial a_{ij}} (\lambda) + C(\lambda) \right)} \quad (15.18) \]
We can apply this to MMIE training, where in this case we get (for uniform word priors)

\[ Z_\lambda = 2^{I_\lambda(q;x)} = \frac{p_\lambda(x|q)}{p_\lambda(x)} = \frac{p_\lambda(x|q)}{\sum_{q'} p_\lambda(x|q')} \]  \hspace{1cm} (15.17)

In discrete case we get update equations:

\[ a_{i,j}^{t+1} = \frac{a_{i,j}^t \left( \frac{\partial \log Z_\lambda}{\partial a_{i,j}} (\lambda) + C(\lambda) \right)}{\sum_{j=1}^{q_i} a_{i,j}^t \left( \frac{\partial \log Z_\lambda}{\partial a_{i,j}} (\lambda) + C(\lambda) \right)} \]  \hspace{1cm} (15.18)

and similar for the other HMM parameters.
Extended Baum-Welch

- Note, that we have lower bound on $C$. 
Extended Baum-Welch

- Note, that we have lower bound on $C$.
- We can prove convergence if $C$ is large enough, but as $C$ gets larger, convergence takes a long time - tradeoff
Extended Baum-Welch

- Note, that we have lower bound on $C$.
- We can prove convergence if $C$ is large enough, but as $C$ gets larger, convergence takes a long time - tradeoff.
- Heuristic: choose least possible value and double it.
Generative vs. Discriminative Modeling of Data
**Other forms of discriminative training**

- Is posterior probability $p(q|x)$ most important thing to optimize?
Other forms of discriminative training

- Is posterior probability $p(q|x)$ most important thing to optimize?
- Bayes Decision Theory says minimum error from:

$$q^*(x) \in \arg\max_q p(q|x)$$

(15.19)
Other forms of discriminative training

- Is posterior probability $p(q|x)$ most important thing to optimize?

- Bayes Decision Theory says minimum error from:

$$q^*(x) \in \arg\max_q p(q|x) \tag{15.19}$$

- But to get $q^*(x)$, we don’t need all the information that exists in the posterior distribution $p(q|x)$, rather we need only the maximum value. Hence, we can approximate the posterior (ask for less information) without error.
Other forms of discriminative training

- Is posterior probability $p(q|x)$ most important thing to optimize?
  - Bayes Decision Theory says minimum error from:
    
    $$q^*(x) \in \arg\max_{q} p(q|x)$$  \hfill (15.19)

- But to get $q^*(x)$, we don’t need all the information that exists in the posterior distribution $p(q|x)$, rather we need only the maximum value. Hence, we can approximate the posterior (ask for less information) without error.

- Ex. approximations to the posterior (for small random $\epsilon$):
  
  $$q^*(x) \in \arg\max_{q}(p(q|x) + \epsilon)$$  \hfill (15.20)
Other forms of discriminative training

- Is posterior probability $p(q|x)$ most important thing to optimize?
- Bayes Decision Theory says minimum error from:

$$q^*(x) \in \arg\max_q p(q|x)$$

(15.19)

- But to get $q^*(x)$, we don’t need all the information that exists in the posterior distribution $p(q|x)$, rather we need only the maximum value. Hence, we can approximate the posterior (ask for less information) without error.

- Ex. approximations to the posterior (for small random $\epsilon$):

$$q^*(x) \in \arg\max_q (p(q|x) + \epsilon)$$

(15.20)

- Since goal is only $q^*(x)$, why not train a model using objective that measures performance based only on $q^*(x)$, and on full posterior?
Other forms of discriminative training

- Is posterior probability $p(q|x)$ most important thing to optimize?
- Bayes Decision Theory says minimum error from:
  \[
  q^*(x) \in \arg\max_q p(q|x) \tag{15.19}
  \]
- But to get $q^*(x)$, we don’t need all the information that exists in the posterior distribution $p(q|x)$, rather we need only the maximum value. Hence, we can approximate the posterior (ask for less information) without error.
- Ex. approximations to the posterior (for small random $\epsilon$):
  \[
  q^*(x) \in \arg\max_q (p(q|x) + \epsilon) \tag{15.20}
  \]
- Since goal is only $q^*(x)$, why not train a model using objective that measures performance based only on $q^*(x)$, and on full posterior?
- Rather than ask for something that is more than what we need (the posterior). Need only find discriminant function that gets low error.
Error training

- This is related to risk minimization with an error-based loss function, but here done for speech.
Error training

- This is related to risk minimization with an error-based loss function, but here done for speech.
- With discriminant function $g_q$, we could produce decision rule:

$$q^*(x) \in \arg\max_q g_q(x|\lambda)$$  \hspace{1cm} (15.21)
Minimum classification error (MCE) training, in speech recognition, has the goal to minimize classification error function directly.
Minimum classification error (MCE) training, in speech recognition, has the goal to minimize classification error function directly.

Approach: Error function is a discrete “counting” function, non-differential, hard to optimize continuous parameter space using this objective.

Example:

$$\max_i g_i = \lim_{\eta \to \infty} \log \left( \frac{1}{N} \sum_j \exp(g_j \eta) \right)^{1/\eta} \quad (15.22)$$

For reasonable sized $\eta$, this is a “nice” function.
MCE training

Minimum classification error (MCE) training, in speech recognition, has the goal to minimize classification error function directly.

Approach: Error function is a discrete “counting” function, non-differential, hard to optimize continuous parameter space using this objective.

Instead, use “smooth” continuous differentiable approximations to functions like “max”, and “sign” with smoothness parameters that in the limit approach the hard versions.
MCE training

- Minimum classification error (MCE) training, in speech recognition, has the goal to minimize classification error function directly.

- Approach: Error function is a discrete “counting” function, non-differential, hard to optimize continuous parameter space using this objective.

- Instead, use “smooth” continuous differentiable approximations to functions like “max”, and “sign” with smoothness parameters that in the limit approach the hard versions.

- Example:

\[
\max_i g_i = \lim_{\eta \to \infty} \log \left[ \frac{1}{N} \sum_j \exp(g_j \eta) \right]^{1/\eta} \tag{15.22}
\]
Minimum classification error (MCE) training, in speech recognition, has the goal to minimize classification error function directly.

Approach: Error function is a discrete “counting” function, non-differential, hard to optimize continuous parameter space using this objective.

Instead, use “smooth” continuous differentiable approximations to functions like “max”, and “sign” with smoothness parameters that in the limit approach the hard versions.

Example:

\[
\max_i g_i = \lim_{\eta \to \infty} \log \left[ \frac{1}{N} \sum_j \exp(g_j \eta) \right]^{1/\eta}
\]  

(15.22)

For reasonable sized \( \eta \), this is a “nice” function.
MCE training

- Uses these smoothing functions to approximate classification error, and then use gradient descent to train.
MCE training

- Uses these smoothing functions to approximate classification error, and then use gradient descent to train.

**Misclassification measure:**

\[
d_i(X) = -g_i(X|\Lambda) + \log \left[ \frac{1}{N - 1} \sum_{j \neq i} \exp(g_i(X|\Lambda) \eta) \right]^{1/\eta}
\]  

(15.23)
MCE training

- Uses these smoothing functions to approximate classification error, and then use gradient descent to train.

- Misclassification measure:

\[
d_i(X) = -g_i(X|\Lambda) + \log \left[ \frac{1}{N-1} \sum_{j \neq i} \exp(g_i(X|\Lambda)\eta) \right]^{1/\eta}
\]  

(15.23)

- For large enough \( \eta \), if this is \( d_i(X) > 0 \), then misclassification occurs when we decide class \( i \).
MCE training

- Uses these smoothing functions to approximate classification error, and then use gradient descent to train.

- Misclassification measure:

\[ d_i(X) = -g_i(X|\Lambda) + \log \left[ \frac{1}{N-1} \sum_{j \neq i} \exp(g_i(X|\Lambda)\eta) \right]^{1/\eta} \]  

(15.23)

- For large enough \( \eta \), if this is \( d_i(X) > 0 \), then misclassification occurs when we decide class \( i \).

- Loss function (measures amount of misclassification):

\[ \ell(d) = \frac{1}{1 + \exp(-\gamma d + \theta)}, \quad \gamma \geq 1 \]  

(15.24)
MCE training

- Uses these smoothing functions to approximate classification error, and then use gradient descent to train.

- Misclassification measure:

\[
 d_i(X) = -g_i(X|\Lambda) + \log \left[ \frac{1}{N-1} \sum_{j \neq i} \exp(g_i(X|\Lambda)\eta) \right]^{1/\eta} \tag{15.23}
\]

- For large enough \( \eta \), if this is \( d_i(X) > 0 \), then misclassification occurs when we decide class \( i \).

- Loss function (measures amount of misclassification):

\[
 \ell(d) = \frac{1}{1 + \exp(-\gamma d + \theta)}, \quad \gamma \geq 1 \tag{15.24}
\]

- If \( d \) is smaller than zero, no loss occurs, but positive \( d \) incurs a loss.
MCE Training

- Final classification performance criterion (for a $X$ of class $i$):

$$\ell(X|\Lambda) = \sum_{i=1}^{M} \ell_i(X|\Lambda) 1(X \in C_i)$$

(15.25)

over training set:

$$L(\Lambda) = E_x\{\ell(X|\Lambda)\}$$

(15.26)
MCE Training

- Final classification performance criterion (for a $X$ of class $i$):

$$
\ell(X|\Lambda) = \sum_{i=1}^{M} \ell_i(X|\Lambda) \mathbf{1}(X \in C_i) \hspace{1cm} (15.25)
$$

over training set:

$$
L(\Lambda) = \mathbb{E}_x \{ \ell(X|\Lambda) \} \hspace{1cm} (15.26)
$$

- This can be trained using gradient descent.
MCE Training

- Final classification performance criterion (for a $X$ of class $i$):

$$\ell(X|\Lambda) = \sum_{i=1}^{M} \ell_i(X|\Lambda) \mathbf{1}(X \in C_i)$$

over training set:

$$L(\Lambda) = E_x\{\ell(X|\Lambda)\}$$

- This can be trained using gradient descent.
- Smoothness parameters are $\eta$ and $\gamma$ but tradeoff exists:
**MCE Training**

- Final classification performance criterion (for a $X$ of class $i$):

$$
\ell(X|\Lambda) = \sum_{i=1}^{M} \ell_i(X|\Lambda) 1(X \in C_i) \tag{15.25}
$$

over training set:

$$
L(\Lambda) = E_x \{ \ell(X|\Lambda) \} \tag{15.26}
$$

- This can be trained using gradient descent.
- Smoothness parameters are $\eta$ and $\gamma$ but tradeoff exists:
  - high-values means good approximation to true discrete error measure, but higher order Taylor terms are significant which means training will not be as good.
MCE Training

- Final classification performance criterion (for a $X$ of class $i$):
  \[ \ell(X|\Lambda) = \sum_{i=1}^{M} \ell_i(X|\Lambda) \mathbf{1}(X \in C_i) \]  
  \[ \text{(15.25)} \]

  over training set:
  \[ L(\Lambda) = E_x \{ \ell(X|\Lambda) \} \]  
  \[ \text{(15.26)} \]

- This can be trained using gradient descent.
- Smoothness parameters are $\eta$ and $\gamma$ but tradeoff exists:
  - high-values means good approximation to true discrete error measure, but higher order Taylor terms are significant which means training will not be as good.
  - Low-values means smooth functions without significant higher-order terms, but poor approximation to true discrete error function.
Discriminative Training

MCE Training

- Final classification performance criterion (for a $X$ of class $i$):

\[
\ell(X|\Lambda) = \sum_{i=1}^{M} \ell_i(X|\Lambda) 1(X \in C_i) \tag{15.25}
\]

over training set:

\[
L(\Lambda) = E_x \{\ell(X|\Lambda)\} \tag{15.26}
\]

- This can be trained using gradient descent.
- Smoothness parameters are $\eta$ and $\gamma$ but tradeoff exists:
  - high-values means good approximation to true discrete error measure, but higher order Taylor terms are significant which means training will not be as good.
  - Low-values means smooth functions without significant higher-order terms, but poor approximation to true discrete error function.
- Generalized probabilistic descent (GPD): given smoothness guarantees (bounded functions of Hessian), we have convergence guarantees of this algorithm.
Discriminative training and computation

- Given training data of $D$ utterances, $\mathcal{D} = \left\{ (x_{1:T_i}^{(i)}, w^{(i)}) \right\}_{i=1}^{D}$. 
Discriminative training and computation

- Given training data of $D$ utterances, $\mathcal{D} = \left\{ (x_{1:T_i}^{(i)}, w^{(i)}) \right\}_{i=1}^{D}$.
- ML training requires computing things like:

$$\log \sum_{q_{1:T} \in Q_{1:T}} p(x_{1:T_i}^{(i)}, q_{1:T} | \lambda)$$

for the numerator (i.e., sum over all paths corresponding to worse sequence). This is hard but doable.
Discriminative training and computation

- Given training data of $D$ utterances, $\mathcal{D} = \left\{ (x^{(i)}_{1:T_i}, w^{(i)}) \right\}^{D}_{i=1}$.
- ML training requires computing things like:
  \[
  \log \sum_{q_{1:T} \in Q_{1:T}(w_{1:N_i})} p(x^{(i)}_{1:T_i}, q_{1:T} | \lambda) \tag{15.27}
  \]
  for the numerator (i.e., sum over all paths corresponding to worse sequence). This is hard but doable.
- Discriminative training requires computing a denominator, which is something of the form:
  \[
  \log \sum_{w_{1:N}} \sum_{q_{1:T} \in Q_{1:T}(w_{1:N})} p(x^{(i)}_{1:T_i}, q_{1:T} | \lambda) = \log p(x^{(i)}_{1:T_i} | \lambda) \tag{15.28}
  \]
Discriminative training and computation

- Given training data of \( D \) utterances, \( \mathcal{D} = \left\{ (x_{1:T_i}^{(i)}, w^{(i)}) \right\}_{i=1}^{D} \).

- ML training requires computing things like:

\[
\log \sum_{q_{1:T} \in \mathcal{Q}_{1:T}} p(x_{1:T_i}^{(i)}, q_{1:T} | \lambda) \quad (15.27)
\]

for the numerator (i.e., sum over all paths corresponding to worse sequence). This is hard but doable.

- Discriminative training requires computing a denominator, which is something of the form:

\[
\log \sum_{w_{1:N}} \sum_{q_{1:T} \in \mathcal{Q}_{1:T}} p(x_{1:T_i}^{(i)}, q_{1:T} | \lambda) = \log p(x_{1:T_i}^{(i)} | \lambda) \quad (15.28)
\]

- This is computationally intractable, and hence we need a way of approximating the denominator.
Observation Densities

There are many possible forms of observation densities.

- continuous observations (often modeled by mixtures)

\[
b_j(x_t) = \sum_{k=1}^{K_j} c_{jk}N(x_t | \mu_{jk}, \Sigma_{jk})
\]

(15.29)

where \(N(x | \mu, \Sigma)\) is a multivariate Gaussian.

Often the Gaussians are diagonal, but a mixture of such densities still represents dependence.

Discrete (vector quantization)

\[
b_j(x_t) = \prod_{i=1}^{K} p_1(x_t = x_{ij})
\]

(15.30)

So called "Semi-continuous" HMMs:

\[
b_j(x) = \sum_{k=1}^{K_j} c_{jk}N(x | \mu_{m(jk)}, \Sigma_{s(jk)})
\]

(15.31)

where \(m\) and \(s\) are mappings. Global sharing of a pool of densities.
There are many possible forms of observation densities.

- continuous observations (often modeled by mixtures)

\[
b_j(x_t) = \sum_{k=1}^{K_j} c_{jk} \mathcal{N}(x_t | \mu_{jk}, \Sigma_{jk})
\]  

(15.29)

where \( \mathcal{N}(x | \mu, \Sigma) \) is a multivariate Gaussian. Often the Gaussians are diagonal, but a mixture of such densities still represents dependence.
Observation Densities

There are many possible forms of observation densities.

- **continuous observations (often modeled by mixtures)**

\[
b_j(x_t) = \sum_{k=1}^{K_j} c_{jk} \mathcal{N}(x_t | \mu_{jk}, \Sigma_{jk})
\]  

(15.29)

where \( \mathcal{N}(x | \mu, \Sigma) \) is a multivariate Gaussian. Often the Gaussians are diagonal, but a mixture of such densities still represents dependence.

- **Discrete (vector quantization)**

\[
b_j(x_t) = \prod_{k=1}^{K} p_{ij}^{1(x_t=x^{(ij)})}
\]  

(15.30)
Observation Densities

There are many possible forms of observation densities.

- **continuous observations (often modeled by mixtures)**
  \[
  b_j(x_t) = \sum_{k=1}^{K_j} c_{jk} N(x_t | \mu_{jk}, \Sigma_{jk})
  \]  
  where \( N(x | \mu, \Sigma) \) is a multivariate Gaussian. Often the Gaussians are diagonal, but a mixture of such densities still represents dependence.

- **Discrete (vector quantization)**
  \[
  b_j(x_t) = \prod_{k=1}^{K} p_{ij}^{1(x_t=x(ij))}
  \]  

- **So called “Semi-continuous” HMMs:**
  \[
  b_j(x) = \sum_{k=1}^{K} c_{jk} N(x | \mu_{m(jk)}, \Sigma_{s(jk)})
  \]  
  where \( m \) and \( s \) are mappings. Global sharing of a pool of densities.
Observation Densities

- Neural networks

\[ b_j(x_t) \propto p(Q_t = j | x_t) / p(Q_t = j) \]  \hspace{1cm} (15.32)

where \( p(Q_t = j | x_t) \) is the probability of state \( j \) given \( x_t \) computed using a neural network or a deep model.
Observation Densities

- Neural networks

\[ b_j(x_t) \propto p(Q_t = j|x_t)/p(Q_t = j) \tag{15.32} \]

where \( p(Q_t = j|x_t) \) is the probability of state \( j \) given \( x_t \) computed using a neural network or a deep model.

- Note that in an HMM, this gives scores that are proportional to scaled likelihoods. i.e., if

\[
\prod_t p(q_t|x_t)p(q_t|q_{t-1}) = \prod_t \frac{p(x_t|q_t)p(q_t)}{p(x_t)} p(q_t|q_{t-1}) \tag{15.33}
\]

\[
= \prod_t \frac{1}{p(x_t)} \prod_t p(x_t|q_t)p(q_t)p(q_t|q_{t-1}) \tag{15.34}
\]
Observation Densities

- **Neural networks**

\[ b_j(x_t) \propto p(Q_t = j|x_t)/p(Q_t = j) \] (15.32)

where \( p(Q_t = j|x_t) \) is the probability of state \( j \) given \( x_t \) computed using a neural network or a deep model.

- **Note** that in an HMM, this gives scores that are proportional to scaled likelihoods. I.e., if

\[
\prod_t p(q_t|x_t)p(q_t|q_{t-1}) = \prod_t \frac{p(x_t|q_t)p(q_t)}{p(x_t)} p(q_t|q_{t-1})
\] (15.33)

\[
= \prod_t \frac{1}{p(x_t)} \prod_t p(x_t|q_t)p(q_t)p(q_t|q_{t-1})
\] (15.34)

- **Thus**, if we use \( b_{q_t}(x_t) = p(q_t|x_t)/p(q_t) \) we get:

\[
\prod_t \frac{1}{p(x_t)} \prod_t p(x_t|q_t)p(q_t)p(q_t|q_{t-1})
\] (15.35)
States

- Some states might be non-emitting (i.e., have no associated observation). For example, start and stop state.
States

- Some states might be non-emitting (i.e., have no associated observation). For example, start and stop state.
- Indicated by concentric circles.
Some states might be non-emitting (i.e., have no associated observation). For example, start and stop state.

Indicated by concentric circles.

we can see this as augmenting the output alphabet with a special symbol indicating a “non-emission”
Rabiner(Moore) vs. Jelinek (Mealy) HMMs

Recall this distinction: Two different ways to draw (and think of) SFSA view of HMMs.
Rabiner (Moore) vs. Jelinek (Mealy) HMMs

Recall this distinction: Two different ways to draw (and think of) SFSA view of HMMs.

Rabiner (Moore) HMM:
- States are circles, transitions are arrows between circles. Observations are associated with the states.
Rabiner(Moore) vs. Jelinek (Mealy) HMMs

Recall this distinction: Two different ways to draw (and think of) SFSA view of HMMs.

Rabiner (Moore) HMM:
- States are circles, transitions are arrows between circles. Observations are associated with the states.

Jelinek (Mealy) HMMs:
- States and observations are associated with edges transitions between circles.
HMMs

- Representations have same representational “capacity”
HMMs

- Representations have same representational “capacity”
- Mealy approach good for representing finite state transducers:
HMMs

- Representations have same representational “capacity”
- Mealy approach good for representing finite state transducers:
  - given you are in a current state, a particular input will cause the probabilistic transition to another state and will cause an output symbol to be emitted (also probabilistically).
HMMs

- Representations have same representational “capacity”
- Mealy approach good for representing finite state transducers:
  - given you are in a current state, a particular input will cause the probabilistic transition to another state and will cause an output symbol to be emitted (also probabilistically).
  - FST are amenable to manipulation (composition, hierarchies, etc.)
HMMs

- Representations have same representational “capacity”
- Mealy approach good for representing finite state transducers:
  - given you are in a current state, a particular input will cause the probabilistic transition to another state and will cause an output symbol to be emitted (also probabilistically).
  - FST are amenable to manipulation (composition, hierarchies, etc.)
  - Word-graphs and lattices (we’ll soon see) use this form of representation.
Recall that for speech, HMM transition matrix graph is often strictly left-to-right (used for sub-word sequences):
Recall that for speech, HMM transition matrix graph is often strictly left-to-right (used for sub-word sequences):

We also covered the duration modeling property of HMMs (that it can be flexible), but do be mindful of the Viterbi score issue when using HMM duration models.
Recall that for speech, HMM transition matrix graph is often strictly left-to-right (used for sub-word sequences):

We also covered the duration modeling property of HMMs (that it can be flexible), but do be mindful of the Viterbi score issue when using HMM duration models.

As we’ve stated, this L2R topology helps discriminability (ability to distinguish one sound from another).
Recall that for speech, HMM transition matrix graph is often strictly left-to-right (used for sub-word sequences):

We also covered the duration modeling property of HMMs (that it can be flexible), but do be mindful of the Viterbi score issue when using HMM duration models.

As we’ve stated, this L2R topology helps discriminability (ability to distinguish one sound from another).

Moreover, we can use it to represent word pronunciations.
What does Markov chain tell us? One thing is the pronunciation of a word. Each state might correspond to a phone.
Word Pronunciations

- What does Markov chain tell us? One thing is the pronunciation of a word. Each state might correspond to a phone.
- “Pronunciation modeling” is an important sub-field within speech recognition.
Word Pronunciations

- What does Markov chain tell us? One thing is the pronunciation of a word. Each state might correspond to a phone.
- “Pronunciation modeling” is an important sub-field within speech recognition.
- Often words have only one pronunciation
The same word, however, can have many pronunciations.
Word Pronunciations

- The same word, however, can have many pronunciations.
- Example: The pronunciation Markov transition matrix for the word “and”
The same word, however, can have many pronunciations.

Example: The pronunciation Markov transition matrix for the word “and”

In general, each word has a set of associated pronunciations, and sometimes there can be many.
Word Pronunciations

- The same word, however, can have many pronunciations.
- Example: The pronunciation Markov transition matrix for the word “and”

In general, each word has a set of associated pronunciations, and sometimes there can be many.

How do we know how a word is pronounced?
Word Pronunciation: pronunciation dictionaries/lexicons

- There are standard pronunciation dictionaries that one can use. E.g., PRONLEX http://www.ldc.upenn.edu/Catalog/readme_files/comlex_pron.readme.html or CMU dict http://www.speech.cs.cmu.edu/cgi-bin/cmudict
∃ standard pronunciation dictionaries that one can use. E.g., PRONLEX http://www.ldc.upenn.edu/Catalog/readme_files/comlex_pron.readme.html or CMU dict http://www.speech.cs.cmu.edu/cgi-bin/cmudict

Examples from pronlex (see above link for details):
bating .xb’et.IG
abba ’@b.x #NAME
abbenhaus ’@b.Inh+Ws #NAME
abbey ’@b.i #NAME
abbott ’@b.xt #NAME
abboud .xb’ud #NAME
abbreviated .xbr’iv.i+et.Id
abby ’@b.i
abdominal .@bd’am.In.xl
Exist standard pronunciation dictionaries that one can use. E.g., PRONLEX http://www.ldc.upenn.edu/Catalog/readme_files/comlex_pron.readme.html or CMU dict http://www.speech.cs.cmu.edu/cgi-bin/cmudict

Examples from pronlex (see above link for details):
- bating .xb’et.IG
- abba ’@b.x #NAME
- abbenhaus ’@b.Inh+Ws #NAME
- abbey ’@b.i #NAME
- abbott ’@b.xt #NAME
- abboud .xb’ud #NAME
- abbreviated .xb’iv.i+et.Id
- abby ’@b.i
- abdominal .@bd’am.In.xl

Commercial ASR systems use their own pronunciation lexicon, and this is a critical part of the performance of such systems.
...some American dialect distinguish the vowels in “sawed” and “sod”, while others do not; the ending “-ing” can be pronounced with a vowel more like “heed” or one more like “hid”, and with a final consonant like that of “sing” or like that of “sin”. This does not take account of considerable variation of actual quality in these sounds: thus some (New Yorkers) pronounce the vowel of “sawed” as a sequence of a vowel like that in “Sue” followed by one like that in “Bud”, while in less stigmatized dialects it is a single vowel (that may or may not be like that in “sod”).
Combining all these variants for the transcription of the word “dogging” we would get 12 pronunciations – three versions of the first vowel, two versions of the second vowel, and two versions of the final consonant. Then someone else comes along to tell us that some Chicagoans not only merge the vowels in “sawed” and “sod” but also move both of them towards the front of mouth, with a sound similar (in extreme cases) to the more standard pronunciation of “sad”. Now we have $4 \times 2 \times 2 = 16$ pronunciations for the simple word “dogging” – with a comparable 16 available for “logging” and “hogging” and so forth, and plenty of variants yet to catalogue.
Word Pronunciation: units

- Pronunciation lexica can be big (100k-500k).
Word Pronunciation: units

- Pronunciation lexica can be big (100k-500k).
- Many possible units may be used to specify a pronunciation
Word Pronunciation: units

- Pronunciation lexica can be big (100k-500k).
- Many possible units may be used to specify a pronunciation
- most common: phonemes and their realizations, phones. Ex: using ARPAbet chocolate pudding → CaKxlIt pUdG
Pronunciation lexica can be big (100k-500k).

Many possible units may be used to specify a pronunciation.

Most common: phonemes and their realizations, phones. Ex: using ARPAbet chocolate pudding → CaKxIlt pUdG.

Phones can often be characterized acoustically (using formants, and their characteristic frequencies).
Word Pronunciation: units

- Pronunciation lexica can be big (100k-500k).
- Many possible units may be used to specify a pronunciation
- Most common: phonemes and their realizations, phones. Ex: using ARPAbet chocolate pudding → CaKxIlt pUdG
- Phones can often be characterized acoustically (using formants, and their characteristic frequencies)
- Syllables - longer (200ms) and more (about 3000 for English)
Word Pronunciation: units

- Pronunciation lexica can be big (100k-500k).
- Many possible units may be used to specify a pronunciation
  - most common: phonemes and their realizations, phones. Ex: using ARPAbet chocolate pudding → CaKxIlIt pUdG
  - phones can often be characterized acoustically (using formants, and their characteristic frequencies)
  - syllables - longer (200ms) and more (about 3000 for English)
  - individual articulatory gestures within the vocal tract (semi-synchronously), very low-level. Factored representation.
Word Pronunciations and Discriminability

- How many pronunciations should a given word model be given?
Word Pronunciations and Discriminability

- How many pronunciations should a given word model be given?
- If a word has $N$ possible pronunciations, we could include all $N$ pronunciations.
Word Pronunciations and Discriminability

- How many pronunciations should a given word model be given?
- If a word has $N$ possible pronunciations, we could include all $N$ pronunciations.
- This is generatively accurate, but can cause confusion with other words.
How many pronunciations should a given word model be given?

If a word has $N$ possible pronunciations, we could include all $N$ pronunciations.

This is generatively accurate, but can cause confusion with other words.

Too few pronunciations per word, poor model.
Word Pronunciations and Discriminability

How many pronunciations should a given word model be given?

If a word has $N$ possible pronunciations, we could include all $N$ pronunciations.

This is generatively accurate, but can cause confusion with other words.

Too few pronunciations per word, poor model.

Too many pronunciations per word, performance (in terms of classification error) drops due to confusability between words.
How many pronunciations should a given word model be given?

If a word has $N$ possible pronunciations, we could include all $N$ pronunciations.

This is generatively accurate, but can cause confusion with other words.

Too few pronunciations per word, poor model.

Too many pronunciations per word, performance (in terms of classification error) drops due to confusability between words.

Ideal point is somewhere in the middle. Where can only be determined empirically.
HMM: Pronunciation Modeling

- Problem with dictionaries is that they typically give "canonical" pronunciations, or what are called "BASEFORMS"
Problem with dictionaries is that they typically give “canonical” pronunciations, or what are called “BASEFORMS”

Words are pronounced in different many ways, depending on context.
Problem with dictionaries is that they typically give “canonical” pronunciations, or what are called “BASEFORMS”.

Words are pronounced in different many ways, depending on context.

Two approaches:
Problem with dictionaries is that they typically give “canonical” pronunciations, or what are called “BASEFORMS”.

Words are pronounced in different many ways, depending on context.

Two approaches:

- introduce many possible pronunciations for each word irrespective of context. But this can increase confusability (words start blurring into each other)
Problem with dictionaries is that they typically give “canonical” pronunciations, or what are called “BASEFORMS”.

Words are pronounced in different many ways, depending on context.

Two approaches:

- Introduce many possible pronunciations for each word irrespective of context. But this can increase confusability (words start blurring into each other).
- Map to correct pronunciation dynamically based on context, and questions about context, only include a small number of pronunciations per context.
Problem with dictionaries is that they typically give “canonical” pronunciations, or what are called “BASEFORMS”.

Words are pronounced in different many ways, depending on context.

Two approaches:

- Introduce many possible pronunciations for each word irrespective of context. But this can increase confusability (words start blurring into each other)
- Map to correct pronunciation dynamically based on context, and questions about context, only include a small number of pronunciations per context. In this case, need need mapping \( T(\text{BASEFORM}) \rightarrow \text{SURFACE FORM} \) where surface form has details about variability.
HMM: Pronunciation Modeling

- Ex: phonemic spelling of bottle:

\[
\begin{align*}
\text{phonemic spelling of bottle:} & = /b\text{aa}t\text{ax}l/ \\
\text{(in ARPABET form)} & = \text{[b aa dx el]}
\end{align*}
\]
HMM: Pronunciation Modeling

- Ex: phonemic spelling of bottle:
- /b aa t ax l/ (in ARPABET form)
HMM: Pronunciation Modeling

- Ex: phonemic spelling of bottle:
- /b a:t a l/ (in ARPABET form)
- In American English, /t/ is “flapped” ⇒ [dx].
HMM: Pronunciation Modeling

- Ex: phonemic spelling of bottle:
  - /b aa t ax l/ (in ARPABET form)
  - In American English, /t/ is “flapped” \( \Rightarrow [dx] \).
  - Also /ax l/ \( \Rightarrow [el] \)
HMM: Pronunciation Modeling

- Ex: phonemic spelling of bottle:
  /b aa t ax l/ (in ARPABET form)
- In American English, /t/ is “flapped” ⇒ [dx].
- Also /ax l/ ⇒ [el]
- Resulting surface form: [b aa dx el]
HMM: Pronunciation Modeling

- Ex: phonemic spelling of bottle:
  - /b aa t ax l/ (in ARPABET form)
- In American English, /t/ is “flapped” ⇒ [dx].
- Also /ax l/ ⇒ [el]
- Resulting surface form: [b aa dx el]
- Goal: build probabilistic mapping from base (phoneme) to surface forms (phones).
HMM: Pronunciation Modeling

- Ex: phonemic spelling of bottle:
  /b aa t ax l/ (in ARPABET form)
- In American English, /t/ is “flapped” ⇒ [dx].
- Also /ax l/ ⇒ [el]
- Resulting surface form: [b aa dx el]
- Goal: build probabilistic mapping from base (phoneme) to surface forms (phones).
- Might also have phoneme deletion, so need to include special “_” in output y alphabet.
HMM: Pronunciation Modeling

- Ex: phonemic spelling of bottle:
  /b aa t ax l/ (in ARPABET form)
- In American English, /t/ is “flapped” ⇒ [dx].
- Also /ax l/ ⇒ [el]
- Resulting surface form: [b aa dx el]
- Goal: build probabilistic mapping from base (phoneme) to surface forms (phones).
- Might also have phoneme deletion, so need to include special “_” in output y alphabet.
- Let $x = x_{1:m}$ be a string of $m$ phonemes (baseforms) and $y = y_{1:m}$ be a string of $m$ phones (surface forms).
HMM: Pronunciation Modeling

- Ex: phonemic spelling of bottle:
  - /b aa t ax l/ (in ARPABET form)
- In American English, /t/ is “flapped” ⇒ [dx].
- Also /ax l/ ⇒ [el]
- Resulting surface form: [b aa dx el]
- Goal: build probabilistic mapping from base (phoneme) to surface forms (phones).
- Might also have phoneme deletion, so need to include special “_” in output y alphabet.
- Let \( x = x_{1:m} \) be a string of \( m \) phonemes (baseforms) and \( y = y_{1:m} \) be a string of \( m \) phones (surface forms).
- Might use a model such as:

\[
p(y|x) = \prod_{n} p(y_n|x, y_{1:n-1}) \tag{15.36}
\]
HMM: Pronunciation Modeling

- Need to make conditional independence assumptions to be tractable.
HMM: Pronunciation Modeling

- Need to make conditional independence assumptions to be tractable. **two typical models:**
HMM: Pronunciation Modeling

- Need to make conditional independence assumptions to be tractable. Two typical models:

1) Current phone is dependent on window of phonemes and previous phone

\[
p(y|x) = \prod_{n} p(y_n|x_{n-r:n+r}, y_{n-1})
\]  

(15.37)
HMM: Pronunciation Modeling

- Need to make conditional independence assumptions to be tractable. Two typical models:

1) Current phone is dependent on window of phonemes and previous phone

\[
p(y|x) = \prod_{n} p(y_{n}|x_{n-r:n+r}, y_{n-1}) \tag{15.37}\]

2) Current phone is dependent only on window of phonemes

\[
p(y|x) = \prod_{n} p(y_{n}|x_{n-r:n+r}) \tag{15.38}\]
HMM: Pronunciation Modeling

Basic Approach:

- obtain canonical transcriptions of language (e.g., via say PRONLEX) for speech training material
HMM: Pronunciation Modeling

Basic Approach:

- obtain canonical transcriptions of language (e.g., via say PRONLEX) for speech training material
- Obtain SURFACE form transcriptions for same speech material (hand transcribed by a phonetician ideally)
HMM: Pronunciation Modeling

Basic Approach:

- obtain canonical transcriptions of language (e.g., via say PRONLEX) for speech training material
- Obtain SURFACE form transcriptions for same speech material (hand transcribed by a phonetician ideally)
- Align the baseforms and surface forms with dynamic programming (using string edit distance)
HMM: Pronunciation Modeling

Basic Approach:

- obtain canonical transcriptions of language (e.g., via say PRONLEX) for speech training material
- Obtain SURFACE form transcriptions for same speech material (hand transcribed by a phonetician ideally)
- Align the baseforms and surface forms with dynamic programming (using string edit distance) — this gives training data pairs, but we do need local phoneme/phone distances, cost constraints, etc. for DP
Basic Approach:

- obtain canonical transcriptions of language (e.g., via say PRONLEX) for speech training material
- Obtain SURFACE form transcriptions for same speech material (hand transcribed by a phonetician ideally)
- Align the baseforms and surface forms with dynamic programming (using string edit distance) — this gives training data pairs, but we do need local phoneme/phone distances, cost constraints, etc. for DP
- Learn $p(y_k | x_{k-r:k+r})$ from this aligned data
HMM: Pronunciation Modeling

Basic Approach:

- obtain canonical transcriptions of language (e.g., via say PRONLEX) for speech training material
- Obtain SURFACE form transcriptions for same speech material (hand transcribed by a phonetician ideally)
- Align the baseforms and surface forms with dynamic programming (using string edit distance) — this gives training data pairs, but we do need local phoneme/phone distances, cost constraints, etc. for DP
- Learn $p(y_k|x_{k-r:k+r})$ from this aligned data
- Use this mapping to transform words as they are hypothesized in the ASR system
HMM: Pronunciation Modeling

Basic Approach:

- obtain canonical transcriptions of language (e.g., via say PRONLEX) for speech training material
- Obtain SURFACE form transcriptions for same speech material (hand transcribed by a phonetician ideally)
- Align the baseforms and surface forms with dynamic programming (using string edit distance) — this gives training data pairs, but we do need local phoneme/phone distances, cost constraints, etc. for DP
- Learn $p(y_k | x_{k-r:k+r})$ from this aligned data
- Use this mapping to transform words as they are hypothesized in the ASR system
- Hence, we get non-dictionary pronunciations, even for words for which we have never seen surface forms
HMM: Pronunciation Modeling

How to learn these models:

- approximately 50 phonemes, 200 phones
HMM: Pronunciation Modeling

How to learn these models:

- approximately 50 phonemes, 200 phones
- context of size \((2r + 1)\)
HMM: Pronunciation Modeling

How to learn these models:

- approximately 50 phonemes, 200 phones
- context of size \((2r + 1)\)
- implies table of size \(200 \times 502^r + 1\)
HMM: Pronunciation Modeling

How to learn these models:

- approximately 50 phonemes, 200 phones
- context of size $(2r + 1)$
- implies table of size $200 \times 502^{r+1}$
HMM: Pronunciation Modeling

How to learn these models:

- approximately 50 phonemes, 200 phones
- context of size \((2r + 1)\)
- implies table of size \(200 \times 502^{r+1}\)

Solution, decision trees (DTs):

- pools together common contexts (e.g., matters not the particular phone, but rather that it is vowel)
HMM: Pronunciation Modeling

How to learn these models:
- approximately 50 phonemes, 200 phones
- context of size \((2r + 1)\)
- implies table of size \(200 \times 502^{r+1}\)

Solution, decision trees (DTs):
- pools together common contexts (e.g., matters not the particular phone, but rather that it is vowel)
- lower-dimensional representation (need less data)
HMM: Pronunciation Modeling

How to learn these models:

- approximately 50 phonemes, 200 phones
- context of size \(2r + 1\)
- implies table of size \(200 \times 502^{r+1}\)

Solution, decision trees (DTs):

- pools together common contexts (e.g., matters not the particular phone, but rather that it is vowel)
- lower-dimensional representation (need less data)
- Generalizes better
HMM: Pronunciation Modeling

How to learn these models:
- approximately 50 phonemes, 200 phones
- context of size \((2^r + 1)\)
- implies table of size \(200 \times 502^{r+1}\)

Solution, decision trees (DTs):
- pools together common contexts (e.g., matters not the particular phone, but rather that it is vowel)
- lower-dimensional representation (need less data)
- Generalizes better

DT input is a list of “features”, and a set of possible questions about these features, such as \(x \in C_1\), meaning is \(x\) a member of the set that has answer “yes” to the question. \(C_1\) could be, say, set of all vowels.
HMM: Pronunciation Modeling

- Leaf node is distribution over all phones.

Each leaf's probability distribution counts that context.

How to build these trees?

Consider distribution over just $y_k$ having entropy:

$$H(Y_k) = -\sum p(y_k) \log p(y_k)$$

(15.39)
HMM: Pronunciation Modeling

- Leaf node is distribution over all phones.

$$p(y_k | \text{tree path})$$

- Each leaf’s probability distribution counts that context.
HMM: Pronunciation Modeling

- Leaf node is distribution over all phones.

\[ p(y_k|\text{tree path}) \]

- Each leaf’s probability distribution counts that context.
- How to build these trees?
HMM: Pronunciation Modeling

- Leaf node is distribution over all phones.

Each leaf’s probability distribution counts that context.

How to build these trees?

Consider distribution over just $y_k$ having entropy:

$$H(Y_k) = -\sum_{y_k} p(y_k) \log p(y_k) \quad (15.39)$$
Consider set (or question) $S$ splitting the data, and corresponding conditional entropies:

\[
H(Y_k|x \in S) = - \sum_{y_k} p(y_k|x \in S) \log p(y_k|x \in S) \tag{15.40}
\]

\[
H(Y_k|x \notin S) = - \sum_{y_k} p(y_k|x \notin S) \log p(y_k|x \notin S) \tag{15.41}
\]
Consider set (or question) $S$ splitting the data, and corresponding conditional entropies:

$$H(Y_k | x \in S) = - \sum_{y_k} p(y_k | x \in S) \log p(y_k | x \in S)$$  \hfill (15.40) 

$$H(Y_k | x \notin S) = - \sum_{y_k} p(y_k | x \notin S) \log p(y_k | x \notin S)$$  \hfill (15.41) 

How likely is “$S$” to be true? “$\#()$” is count function.

$$p(x \in S) = \frac{\#(x \in S)}{\#(x \in S) + \#(x \notin S)}$$  \hfill (15.42)
Consider set (or question) $S$ splitting the data, and corresponding conditional entropies:

\[
H(Y_k|x \in S) = -\sum_{y_k} p(y_k|x \in S) \log p(y_k|x \in S) \tag{15.40}
\]

\[
H(Y_k|x \notin S) = -\sum_{y_k} p(y_k|x \notin S) \log p(y_k|x \notin S) \tag{15.41}
\]

- How likely is “$S$” to be true? “#$()$” is count function.

\[
p(x \in S) = \frac{#(x \in S)}{#(x \in S) + #(x \notin S)} \tag{15.42}
\]

- Average entropy after the split:

\[
H(Y_k|S) = H(Y_k|x \in S)p(x \in S) + H(Y_k|x \notin S)p(x \notin S) \tag{15.43}
\]
Consider set (or question) $S$ splitting the data, and corresponding conditional entropies:

$$H(Y_k | x \in S) = - \sum_{y_k} p(y_k | x \in S) \log p(y_k | x \in S)$$  \hspace{1cm} (15.40)

$$H(Y_k | x \notin S) = - \sum_{y_k} p(y_k | x \notin S) \log p(y_k | x \notin S)$$  \hspace{1cm} (15.41)

How likely is “$S$” to be true? “#()” is count function.

$$p(x \in S) = \frac{\#(x \in S)}{\#(x \in S) + \#(x \notin S)}$$  \hspace{1cm} (15.42)

Average entropy after the split:

$$H(Y_k | S) = H(Y_k | x \in S)p(x \in S) + H(Y_k | x \notin S)p(x \notin S)$$  \hspace{1cm} (15.43)

Entropy reduction is the “value” of the split $S$:

$$H(Y_k) - H(Y_k | S) = I(Y_k ; S)$$  \hspace{1cm} (15.44)
Consider all possible splits (sets): $S_1, S_2, \ldots, S_M$. 
Consider all possible splits (sets): $S_1, S_2, \ldots, S_M$.

Can use greedy algorithm (top down) to define tree:
Consider all possible splits (sets): $S_1, S_2, \ldots, S_M$.

Can use greedy algorithm (top down) to define tree:

A) Start with single node top of tree
Consider all possible splits (sets): $S_1, S_2, \ldots, S_M$.

Can use greedy algorithm (top down) to define tree:

A) Start with single node top of tree

B) Then, for each leaf node:
Consider all possible splits (sets): $S_1, S_2, \ldots, S_M$.

Can use greedy algorithm (top down) to define tree:

A) Start with single node top of tree

B) Then, for each leaf node:
   1. 
      compute $I(Y_k; S_i)$ for all splits $I()$ for each node
Consider all possible splits (sets): $S_1, S_2, \ldots, S_M$.

Can use greedy algorithm (top down) to define tree:

A) Start with single node top of tree

B) Then, for each leaf node:
   1. compute $I(Y_k; S_i)$ for all splits $I()$ for each node
   2. If largest $I(Y_k; S_i)$ is large enough, split data down two branches of tree, or if $I()$ is small, stop.
Consider all possible splits (sets): $S_1, S_2, \ldots, S_M$.

- Can use greedy algorithm (top down) to define tree:

A) Start with single node top of tree

B) Then, for each leaf node:

1. compute $I(Y_k; S_i)$ for all splits $I()$ for each node
2. If largest $I(Y_k; S_i)$ is large enough, split data down two branches of tree, or if $I()$ is small, stop.
3. go back to step 1
HMM: Pronunciation Modeling

- Consider all possible splits (sets): $S_1, S_2, \ldots, S_M$.
- Can use greedy algorithm (top down) to define tree:
  
  **A)** Start with single node top of tree
  
  **B)** Then, for each leaf node:
    1. compute $I(Y_k; S_i)$ for all splits $I()$ for each node
    2. If largest $I(Y_k; S_i)$ is large enough, split data down two branches of tree, or if $I()$ is small, stop.
    3. go back to step 1

- But how to choose the sets $S_i$?
HMM: Pronunciation Modeling

- Consider all possible splits (sets): $S_1, S_2, \ldots, S_M$.
- Can use greedy algorithm (top down) to define tree:

  A) Start with single node top of tree
  
  B) Then, for each leaf node:
     1. compute $I(Y_k; S_i)$ for all splits $I()$ for each node
     2. If largest $I(Y_k; S_i)$ is large enough, split data down two branches of tree, or if $I()$ is small, stop.
     3. go back to step 1

- But how to choose the sets $S_i$?

- Input vector $x = x_{k-r:k+r}$ need to ask questions like: “is $x \in S$?” for $S \in 2^X$. 

---

Prof. Jeff Bilmes  
Consider all possible splits (sets): \( S_1, S_2, \ldots, S_M \).

Can use greedy algorithm (top down) to define tree:

A) Start with single node top of tree

B) Then, for each leaf node:
   1. compute \( I(Y_k; S_i) \) for all splits \( I(\cdot) \) for each node
   2. If largest \( I(Y_k; S_i) \) is large enough, split data down two branches of tree, or if \( I(\cdot) \) is small, stop.
   3. go back to step 1

But how to choose the sets \( S_i \)?

Input vector \( x = x_{k-r:k+r} \) need to ask questions like: “is \( x \in S \)” for \( S \in 2^X \).

but \( 2^{|X|} \) elements in power set, or \( 2^{(2r+1)L} \) where \( L \) is number of phonemes
Solution: prior-cluster the phonemes into list of attributes.
HMM: Pronunciation Modeling

- Solution: prior-cluster the phonemes into list of attributes.
- E.g., phone ≡ (consonant manner, consonant place, vowel manner, vowel place)
Solution: prior-cluster the phonemes into list of attributes.

E.g., phone ≡ (consonant manner, consonant place, vowel manner, vowel place)

Each attribute takes on only small (e.g., 12) number of values (which includes n/a)
HMM: Pronunciation Modeling

- Solution: prior-cluster the phonemes into list of attributes.
- E.g., phone \( \equiv \) (consonant manner, consonant place, vowel manner, vowel place)
- Each attribute takes on only small (e.g., 12) number of values (which includes n/a)
- Ask question only of form \((CM, CP, VM, VP) = (4, 3, 1, 2)\), so only \(12^4\) sets per phoneme (still many, so more prior-cluster may be necessary depending on availability of computational resources and training data).
HMM: Pronunciation Modeling

- Solution: prior-cluster the phonemes into list of attributes.
- E.g., phone ≡ (consonant manner, consonant place, vowel manner, vowel place)
- Each attribute takes on only small (e.g., 12) number of values (which includes n/a)
- Ask question only of form \((CM, CP, VM, VP) = (4, 3, 1, 2)\), so only \(12^4\) sets per phoneme (still many, so more prior-cluster may be necessary depending on availability of computational resources and training data).
- There are many other advanced pronunciation modeling techniques, data issues, incorporating rate of speech information, and so on. This is still active area of research.