Logistics

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.

HMMs - Review

- There are many aspects of training data, and collecting training data.
- It is not as simple as just having a string of words — rich transcription, the labels are really a string of mixed continuous/discrete vectors (in most general case).
- In most frequent case, we just have a string of words, w/o word boundaries.
- Learning the HMM parameters using the EM algorithm (for maximum likelihood training).
Outline of today


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
HMMs and word sequences

- HMM distribution $p(x_{1:T}, q_{1:T})$ but what do states mean?
- In speech, we have a hierarchy:
  1. Sentences are composed of words.
  2. Words are composed of pronunciations, and can have multiple pronunciations per word.
  3. Word pronunciations are composed of phones.
  4. Phones are composed of subphones (e.g., triphones).
  5. Triphones are composed of low-level HMM states (triphone states).
- We could treat this as a graphical model, where we have distinct random variables for each of the above levels.
- In many HMM systems, however, the above hierarchy is “flattened” down into a single state variable and state sequence $q_{1:T}$.
- Hence, all of this needs somehow to be encoded in $q_{1:T}$.

In reality, a given state $q \in D_Q$ encodes all levels of the hierarchy simultaneously — $\exists$ a one-to-one mapping that (perhaps implicitly) performs the following relation:

$$q_{1:T} \leftrightarrow [\left( w_1, \ldots, w_N \right), \left( r_1, \ldots, r_N \right),$$

$$\left( (h_{1,1}, \ldots, h_{1,\ell_1}), (h_{2,1}, \ldots, h_{2,\ell_2}), \ldots, (h_{N,1}, \ldots, h_{N,\ell_N}) \right),$$

$$\left( (h'_{1,1}, \ldots, h'_{1,\ell_1}), (h'_{2,1}, \ldots, h'_{2,\ell_2}), \ldots, (h'_{N,1}, \ldots, h'_{N,\ell_N}) \right),$$

$$\left( (\vec{y}_{1,1}, \ldots, \vec{y}_{1,\ell_1}), (\vec{y}_{2,1}, \ldots, \vec{y}_{2,\ell_2}), \ldots, (\vec{y}_{N,1}, \ldots, \vec{y}_{N,\ell_N}) \right)]$$

where $w_1, \ldots, w_N$ is a sequence of words, $r_1, \ldots, r_N$ is a sequence of word pronunciation ids, $h_{i,1}, \ldots, h_{i,\ell_i}$ is a sequence of phones for word pronunciation $i$, $h'_{i,1}, \ldots, h'_{i,\ell_i}$ is a sequence of subphones for word pronunciation $i$, and $\vec{y}_{i,j}$ is a ordered vector of states (e.g., for a tri-phone, three states, beginning/middle/end of the tri-phone).
HMMs and word sequences

- Hence, for a given word sequence \( w_1, \ldots, w_N \) there can be many possible states compatible with that word sequence.
- Let the states compatible with word sequence \( w_1, \ldots, w_N \) be represented as: \( Q_{1:T}(w_{1:N}) \). Then we might wish to form:

\[
p(x_{1:T}, w_{1:N}) = \sum_{q_{1:T} \in Q_{1:T}(w_{1:N})} p(x_{1:T}, q_{1:T} | \lambda)
\]

and this corresponds to the joint probability of the observations \( x_{1:T} \) and the word sequence.

HMMs, word sequences, and ML training

- Given training data of \( D \) utterances, \( \mathcal{D} = \{(x_{1:T_i}^{(i)}, w^{(i)})\}_{i=1}^{D} \), where
  - \( x_{1:T_i}^{(i)} \) is a matrix of speech features and
  - \( w^{(i)} = (w_1^{(i)}, w_2^{(i)}, \ldots, w_{N_i}^{(i)}) = w_{1:N_i}^{(i)} \) is a length \( N_i \) sequence of word labels (transcription) of the speech.
- Maximum likelihood training, then becomes:

\[
\lambda^* \in \arg \max_{\lambda} \log \prod_{i=1}^{D} \sum_{q_{1:T} \in Q_{1:T}(w_{1:N_i}^{(i)})} p(x_{1:T_i}^{(i)}, q_{1:T} | \lambda)
\]

- EM training, and the auxiliary function \( Q(\lambda, \lambda^g) \) then needs to do this restricted summation \( \sum_{q_{1:T} \in Q_{1:T}(w_{1:N_i}^{(i)})} \) for each utterance. I.e.,

\[
Q(\lambda, \lambda^g) = \sum_i \sum_{q_{1:T} \in Q_{1:T}(w_{1:N_i}^{(i)})} p_{\lambda^g}(q_{1:T} | x_{1:T_i}^{(i)}) \log p_{\lambda}(x_{1:T_i}^{(i)}, q_{1:T})
\]
EM isn’t the only way to learn parameters. We can instead, for example, use stochastic gradient descent methods:

Suppose we wanted to use a gradient descent like algorithm on $f(\lambda) = \log p(x_{1:T}|\lambda)$, as in

$$
\frac{\partial}{\partial \lambda} f(\lambda) = \frac{\partial}{\partial \lambda} \log p(x_{1:T}|\lambda) = \frac{\partial}{\partial \lambda} \log \sum_{q_{1:T}} p(x_{1:T}, q_{1:T}|\lambda) \tag{14.4}
$$

$$
= \frac{\partial}{\partial \lambda} \sum_{q_{1:T}} p(x_{1:T}, q_{1:T}|\lambda) = \frac{\partial}{\partial \lambda} \frac{\sum_{q_{1:T}} p(x_{1:T}, q_{1:T}|\lambda)}{p(x_{1:T}|\lambda)} \tag{14.5}
$$

Say we’re interested in $\partial/\partial a_{ij}$. Let’s expand the numerator above:

$$
\text{numerator} = \frac{\partial}{\partial a_{ij}} \sum_{q_{1:T}} p(x_{1:T}, q_{1:T}|\lambda) = \frac{\partial}{\partial a_{ij}} \sum_{q_{1:T}} \prod_t p(x_t|q_t)p(q_t|q_{t-1}) \tag{14.6}
$$

Define $\mathcal{T}_{ij}(q_{1:T}) \triangleq \{t : q_{t-1} = i, q_t = j\}$, the set of time points where the state sequence ends at $j$ after being in $i$, in the following:

$$
\text{numerator} = \frac{\partial}{\partial a_{ij}} \sum_{q_{1:T}} \prod_t p(x_t|q_t) \prod_{t \in \mathcal{T}_{ij}(q_{1:T})} a_{ij} \prod_{t \notin \mathcal{T}_{ij}(q_{1:T})} p(q_t|q_{t-1}) \tag{14.7}
$$
HMM - learning with gradient descent

We can derive a relatively simple expression for the gradient:

\[
\begin{align*}
\text{num} &= \sum_{q_1:T} \prod_{t} p(x_t|q_t) \frac{\partial}{\partial a_{ij}} a_{ij}^{\mathbb{T}_i(q_1:T)} \prod_{t \not\in \mathbb{T}_i(q_1:T)} p(q_t|q_{t-1}) \\
&= \sum_{q_1:T} \prod_{t} p(x_t|q_t) \frac{|\mathbb{T}_i(q_1:T)|}{a_{ij}} - 1 \prod_{t \not\in \mathbb{T}_i(q_1:T)} p(q_t|q_{t-1}) \\
&= \sum_{q_1:T} \prod_{t} p(x_t|q_t) p(q_t|q_{t-1}) \frac{|\mathbb{T}_i(q_1:T)|}{a_{ij}} = \sum_{q_1:T} p(x_1:T, q_1:T) \frac{|\mathbb{T}_i(q_1:T)|}{a_{ij}} \\
&= \frac{1}{a_{ij}} \sum_{q_1:T} p(x_1:T, q_1:T) \sum_t 1\{q_{t-1} = i, q_t = j\} \\
&= \frac{1}{a_{ij}} \sum_t \sum_{q_1:T} p(x_1:T, q_1:T) 1\{q_{t-1} = i, q_t = j\} \\
&= \frac{1}{a_{ij}} \sum_t p(x_1:T, q_{t-1} = i, q_t = j) \\
\end{align*}
\]

Also need a projection/re-normalization step, to ensure \(\sum_j a_{ij} = 1\).

For gradient descent learning (like EM) we need for all \(t\) the queries \(p(Q_t = j, Q_{t-1} = i | x_{1:T})\) from the HMM. A similar analysis shows that we also need \(\forall t\) \(p(Q_t = i | x_{1:T})\). These are also needed when performing discriminative training.

\[
\frac{\partial}{\partial \lambda} f(\lambda) = \frac{\partial}{\partial \lambda} \sum_{q_1:T} p(x_1:T, q_1:T | \lambda) = \frac{1}{a_{ij}} \sum_t p(x_1:T, q_{t-1} = i, q_t = j) \\
= \frac{1}{a_{ij}} \sum_t p(q_{t-1} = i, q_t = j | x_{1:T}) \\
\]

(14.8)

(14.9)

And a gradient step update, with learning rate \(\alpha\), of the form:

\[
a_{ij} \leftarrow a_{ij} + \alpha \frac{1}{a_{ij}} \sum_t p(q_{t-1} = i, q_t = j | x_{1:T}) \\
\]

(14.10)
On ML training and generative models

- Given training set $D = \{(x^{(i)}_{1:T_i}, w^{(i)})\}_{i=1}^D$, maximum likelihood training adjusts the HMM parameters so that the data is probable (make the data look good).

$$\max_\lambda p(x_{1:T} | \lambda) = \max_\lambda \sum_{q_{1:T}} p(x_{1:T}, q_{1:T} | \lambda)$$  (14.11)

- Maximum likelihood training: make the parameters such that if we were to sample from an HMM, then any data instance that was part of training data would be a “likely” as possible sample, to the extent possible in an HMM.
- An HMM is known as a generative model. It is a model of $p(x|q)p(q)$, in that we can sample a $q$, and then sample an $x$ given $q$, to generate a sample.
- Maximum likelihood training is form of generative training — objective is optimized when the model generates well, given the constraints of the model (i.e., the HMM’s factorization constraints).

Generative vs. Discriminative Modeling of Data

- Big 2D Gaussian mixture

$$p(x) = \sum_i p(x|i)p(i)$$

where $p(x|i) = \mathcal{N}(\mu_i, \Sigma_i)$
Generative vs. Discriminative Modeling of Data

- Five classes, with decision boundaries shown.
- Can model a set of 5 classes with a mixture of mixtures. That is \( p(x) = \sum_{q=1}^{5} \sum_{i_q} p(x|i_q,c)p(i_q,q) \)
- Class-specific rich generative distribution \( p(x|q) = \sum_{i_q} p(x|i_q,q)p(i_q|q) \)
- Complexity exists both within & between classes.
- When goal is classification, why model within-class complexity?

When goal is only to distinguish between classes, need not model within class complexity.

Discriminative training’s goal: produce model that represents only between class boundaries precisely, within class complexity is unnecessary to represent.

Within-class complexity can even be mostly uniform!
Discriminative training of generative models

- Ideal “true” distribution is $p(x, q)$ and ideal (Bayes) decision, given unknown $\bar{x}$, is given by:

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

(14.12)

- Discriminative training of generative models: given generative model $p_\lambda(x|q)$ and $p_\lambda(q)$, adjust parameters so that the decision process:

$$\tilde{q}^* \in \arg \max_q p_\lambda(x|q)p_\lambda(q)$$

(14.13)

makes same decisions as Equation 14.12

- While still generative model, no reason to model within-class complexity.

Discriminant functions

- Accurate decision is made based only on $p(q|x)$.

- But really, for making correct decision, this is more than necessary.

- We don’t need the probability $p(q|x)$ and probability $p(q'|x)$ for all $q' \neq q$. Rather, need only accurate discriminant function $g(q; x)$ in the sense:

$$\arg \max_q p(q|x) = \arg \max_q g(q; x)$$

(14.14)
Accuracy Objectives

In decreasing order of complexity, “accuracy” might ask for
- generative
- posterior
- rank
- max

Discriminative accuracy of a generative model

- How to measure quality of generative model \( \{p_\lambda(x|q), p_\lambda(q)\} \)?
- Accurate in joint distribution: I.e., small KL divergence
  \[
  \text{KL}(p(x,q) \| p_\lambda(x|q)p_\lambda(q))
  \]
  (14.15)
- Accurate in conditional distribution: I.e., small KL divergence
  \[
  \text{KL}(p(q|x) \| \frac{p_\lambda(x|q)p_\lambda(q)}{\sum_q p_\lambda(x|q)p_\lambda(q)})
  \]
  (14.16)
- Rank accurate (we don’t cover this case further here).
- Accurate in decisions: I.e.,
  \[
  \arg\max_q p_\lambda(x|q)p_\lambda(q) = \arg\max_q p(q|x)
  \]
  (14.17)
- In each case, we still can ask for a generative distribution (like an HMM) but the objective for optimization might be different.
Speech recognition research has been at the forefront of discriminative accuracy for generative models (HMMs in particular) for many years (since the late 1980s, e.g., early work, see Peter Brown’s “The Acoustic-Modeling Problem in Automatic Speech Recognition”, 1987).

Objectives for discriminative training of a generative model

Given training data $\mathcal{D} = \{(x^{(i)}, q^{(i)})\}_i$, optimization objective could have many forms, a few of them include;

- **Posterior accuracy**: If $p_\lambda(q|x) = p_\lambda(x|q)p_\lambda(q)/p_\lambda(x)$, then

  $$f_1(\lambda) = \text{KL}(p(q|x)\|p_\lambda(q|x)) \quad (14.18)$$

  requires knowing true $p(q|x)$. Then we find $\min_\lambda f_1(\lambda)$.

- **Empirical approximation of the above based on training data**: maximum conditional likelihood:

  $$f_{\text{emp}}(\lambda) = \frac{1}{N} \sum_{i=1}^N \log p_\lambda(q^{(i)}|x^{(i)}) \quad (14.19)$$

  Then we find $\max_\lambda f_2(\lambda)$. 
Objectives for discriminative accuracy of generative model

- Classification error (risk): If \( p_\lambda(q|x) = p_\lambda(x|q)p_\lambda(q)/p_\lambda(x) \), then

\[
 f_3(\lambda) = \int p(x) \mathbf{1}\left( \arg\max_q p(q|x) \neq \arg\max_q p_\lambda(q|x) \right) \tag{14.20}
\]

requires knowing true \( p(q|x) \). Then we find \( \min_\lambda f_3(\lambda) \).

- Empirical risk minimization:

\[
 f_4(\lambda) = \sum_i \mathbf{1}\left( q^{(i)} \neq \arg\max_q p_\lambda(q|x^{(i)}) \right) \tag{14.21}
\]

Then we find \( \min_\lambda f_4(\lambda) \).

Posterior Accuracy → MMIE

- Consider the posterior accuracy case above:

\[
 \min_\lambda \text{KL}(p(q|x)\|p_\lambda(q|x)) \tag{14.22}
\]

- We have that:

\[
 \text{KL}(p(q|x)\|p_\lambda(q|x)) = \sum_{q,x} p(q,x) \log \frac{p(q|x)}{p_\lambda(q|x)} \tag{14.23}
\]

\[
 = \sum_{q,x} p(q,x) \log p(q|x) - \sum_{q,x} p(q,x) \log p_\lambda(q|x)
\]

\[
 = -H(Q|X) - \sum_{q,x} p(q,x) \log p_\lambda(q|x)
\]

- This is thus the same as (assuming we optimize \( \lambda \) unrelated to \( q \)):

\[
 \arg\max_\lambda \sum_{q,x} p(q,x) \log p_\lambda(q|x) = \arg\max_\lambda \left( \sum_{q,x} p(q,x) \log p_\lambda(q|x) + H(Q) \right)
\]
Continuing:

\[
\sum_{q,x} p(q, x) \log p_\lambda(q|x) + H(q) = \sum_{q,x} p(q, x) \log \frac{p_\lambda(q|x)}{p(q)}
\]

\[
= \sum_{q,x} p(q, x) \log \frac{p_\lambda(q|x)p(x)}{p(q)p(x)}
\]

\[
= \sum_{q,x} p(q, x) \log \frac{p_\lambda(x|q)}{\sum_q p_\lambda(x|q)p(q)}
\]

\[
= I_\lambda(Q; X)
\]

where \(I_\lambda(Q; X)\) is the mutual information between \(X\) and \(Q\) under joint model \(p_\lambda(q|x)p(x)\), where \(p(x) = \sum_q p_\lambda(x|q)p(q)\).

We relate this back to the training data using the weak law of large numbers, which says that if we have enough training data, the above will be approximated.

We get:

\[
I_\lambda(M; X) = \sum_{q,x} p(q, x) \log \frac{p_\lambda(x|q)}{\sum_q p_\lambda(x|q)p(q)}
\]

\[
= \lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^N \log \frac{p_\lambda(x^{(i)}|q^{(i)})}{\sum_q p_\lambda(x^{(i)}|q)p(q)}
\]

(14.24)

where \((x^{(i)}, q^{(i)}) \sim p(x, q)\).

If \(N\) is finite but large, then this is a close approximation, and we get an empirical training function:

\[
I_\lambda(Q; X) = \frac{1}{N} \sum_{i=1}^N \log \frac{p_\lambda(x^{(i)}|q^{(i)})}{\sum_q p_\lambda(x^{(i)}|q)p(q)}
\]

(14.25)

(14.26)
As in MLE, MMIE is a method to train a model based on data.

So parameters are based on large mutual information, but how do we train using $I_\lambda(Q; X)$?


$$f_{\text{MLE}}(x|q) = \log p_\lambda(x|q)$$ \hspace{1cm} (14.27)

and

$$f'_{\text{MLE}}(x|q) = \frac{\partial}{\partial \lambda_i} \log p_\lambda(x|q) = \frac{1}{p_\lambda(x|q)} \frac{\partial}{\partial \lambda_i} p_\lambda(x|q)$$ \hspace{1cm} (14.28)

We know how to compute this as seen earlier today.

For MMIE,

$$f_{\text{MMIE}}(q|x) = \log p_\lambda(x|q) - \log p_\lambda(x)$$ \hspace{1cm} (14.29)

so

$$\frac{\partial}{\partial \lambda_i} f_{\text{MMIE}}(q|x) = f'_{\text{MLE}}(x|q) \left[ 1 - p_\lambda(x|q) \frac{p_\lambda(q)}{p_\lambda(x)} \right]$$

$$- \sum_{q' \neq q} f'_{\text{MLE}}(x|q') p_\lambda(x|q') \frac{p_\lambda(q')}{p_\lambda(x)}$$ \hspace{1cm} (14.30)

$$= f'_{\text{MLE}}(x|q) - \sum_{q'} f'_{\text{MLE}}(x|q') p_\lambda(x|q') \frac{p_\lambda(q')}{p_\lambda(x)}$$ \hspace{1cm} (14.31)

$$= f'_{\text{MLE}}(x|q) - \sum_{q'} f'_{\text{MLE}}(x|q') p_\lambda(q'|x)$$ \hspace{1cm} (14.32)
MMIE: Maximum Mutual Information Estimation

- MMIE gradient has terms that want to go in direction of MLE gradient (based on true word), but also a term that wants to go different direction based on each incorrect word sequence.
- We don’t want the model to highly score incorrect word sequence, so we move model away from doing so.
- We decrease score $p(x|q)$ provided to words that are not $q$.
- Gradient descent is a reasonable way to optimize parameters. But:
  1) no convergence guarantees (not convex); 2) no convergence rate guarantees; 3) very expensive; 4) only first order (higher order, Newton might be better).
- Very expensive since we need to sum over all $q$ (including all possible word sequences!!).

Extended Baum-Welch

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

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

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:

\[ P(\Lambda) = P(\{\Lambda_{ij}\}), i = 1 \ldots p; j = 1 \ldots q_i \]  
\[ (14.34) \]

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

\[ \Lambda \in \triangle = \left\{ \lambda_{ij} : \lambda_{ij} \geq 0, \sum_{j=1}^{q_i} \lambda_{ij} = 1 \right\} \]
\[ (14.35) \]

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

\[ R(\Lambda) = S_1(\Lambda)/S_2(\Lambda) \]
\[ (14.36) \]

How is it that our problem is in this form?

\[ 2I_\lambda(q;x) = \frac{p_\lambda(x|q)}{p_\lambda(x)} = \frac{p_\lambda(x|q)}{\sum_{q'} p(q')p_\lambda(x|q')} \]
\[ (14.37) \]

and for an HMM

\[ p_\lambda(x_1:T|q) = \sum_{q_1:T} \prod_{t} p_\lambda(x_t|q_t)p_\lambda(q_t|q_{t-1}) \]
\[ (14.38) \]

- 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

Extended Baum-Welch

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 14.5.1 (Baum67)

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

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

(14.39)

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

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

(14.40)

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

### 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:

$$P_\lambda(\Lambda) \triangleq S_1(\Lambda) - R(\lambda)S_2(\Lambda)$$  \hspace{1cm} (14.41)

- 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:

$$P_\lambda(\xi) = S_1(\xi) - R(\lambda)S_2(\xi) > 0$$  \hspace{1cm} (14.42)

- 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
  \[ P'(\Lambda) \triangleq P(\Lambda) + C(\Lambda) \]  
  (14.43)

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

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

(14.44)

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
  \[ P''(\Psi) = \Psi_{p+1,1}^d P'(\{\Psi_{ij}\} / \Psi_{p+1,1}) \]  
  (14.45)

which is variable substitution with:

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

and constraint \( \Psi_{p+1,1} = 1 \)  
(14.46)

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

(14.47)

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

- Note, with \( q_{p+1} = 1 \) means that \( \psi_{p+1,1} = 1 \).
Extended Baum-Welch

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 14.5.2 (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 $R()$.

$$[T^C(\lambda)]_{ij} = \frac{\lambda_{ij} \left( \frac{\partial P_{\lambda}}{\partial \lambda_{ij}} (\lambda) + C \right)}{\sum_{j=1}^{q_i} \lambda_{ij} \left( \frac{\partial P_{\lambda}}{\partial \lambda_{ij}} (\lambda) + C \right)}$$ (14.48)

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 $\lambda$. 
MMIE training and growth transformations

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

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

(14.49)

- In discrete case we get update equations:

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

(14.50)

and similar for the other HMM parameters.

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.
Other forms of discriminative training

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

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

(14.51)

- 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 \( \epsilon \)):

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

(14.52)

- 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.
- With discriminant function $g_q$, we could produce decision rule:

$$q^*(x) = \arg\max_q g_q(x|\lambda)$$

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:

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

- 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.
- Misclassification measure:
  \[
  d_i(X) = -g_i(X|\Lambda) + \log \left[ \frac{1}{N-1} \sum_{j \neq i} \exp(g_i(X|\Lambda)\eta) \right]^{1/\eta}
  \]  
  (14.55)
- 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):
  \[
  \ell(d) = \frac{1}{1 + \exp(-\gamma d + \theta)}, \quad \gamma \geq 1
  \]  
  (14.56)
- 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 \(i\)):
  \[
  \ell(X|\Lambda) = \sum_{i=1}^{M} \ell_i(X|\Lambda)1(X \in C_i)
  \]  
  (14.57)
  over training set:
  \[
  L(\Lambda) = E_x \{ \ell(X|\Lambda) \}
  \]  
  (14.58)
- This can be trained using gradient descent.
- Smoothness parameters are \(\eta\) and \(\lambda\) 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 mean 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, $\mathcal{D} = \left\{ (x_{1:T_i}^{(i)}, w^{(i)}) \right\}_{i=1}^D$.
- ML training requires computing things like:

$$
\log \sum_{q_1:T \in \mathcal{Q}_1:T} p(x_{1:T_i}^{(i)}, q_1:T | \lambda) 
$$

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:

$$
\log \sum_{w_1:N} \sum_{q_1:T \in \mathcal{Q}_1:T} p(x_{1:T_i}^{(i)}, q_1:T | \lambda) = \log p(x_{1:T_i}^{(i)} | \lambda) 
$$

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

### 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
Word Pronunciations

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

![Diagram of pronunciation Markov transition matrix]

- 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

- ∃ 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

- Commercial ASR systems use their own pronunciation lexicon, and this is a critical part of the performance of such systems.
Word Pronunciation: pronunciation dictionaries/lexicons

Complex data-intensive process to get this right. Quoting from pronlex

... 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).
- 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)
- 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?
- 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.
DTW (and DP) is one early method people used to recognize speech, and is based on templates.