Logistics

Visit the URL links that were covered in previous lectures.

• Only 1/2 hour office hours today, starting at 3:30pm.
First Guest Lecture, this Thursday, May 16th

Dr. Alex Acero, from Microsoft research, will be giving a guest lecture this Thursday. The abstract follows:

Speech Coding and Enhancement
Dr. Alex Acero
Microsoft Research
Thursday, May 16th, 2013, 1:30-3:20pm, Thomson Hall, 235

In this lecture we’ll cover the fundamentals of speech coding and speech enhancement. Speech coding is used to compress the digital signal for efficient transmission and is used in cellphones and all internet transmissions. Speech enhancement deals with acquiring and restoring a signal in the presence of noise using techniques such as microphone arrays, echo cancellation and single microphone noise suppression.

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”).
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).

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)
- Spinx - 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 20th: 11:45pm: project proposal (1 page max).
- 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 10:00am in the morning.
Office hours and/or email if you have any questions.
HMMs - Review

- HMM properties
- HMMs are not i.i.d., in an HMM nothing is independent of anything else.
- Rather there are conditional independence properties.
- HMM - exponentially decreasing correlation over time.
- State durations and flexible duration models with tied states.
- State duration and Viterbi, renders duration model ineffective.
- State duration and Viterbi, can improve discriminability.
- Jelinek vs. Rabiner HMMs.
- Forms of training data (more on this today).

Outline of today

- On speech training data.
- Training Hidden Markov Models.
HMM Training: Forms of Data

- Training data $\mathcal{D} = \left\{ (x^{(i)}_{1:T_i}, w^{(i)}) \right\}_i$, where $x^{(i)}_{1:T_i}$ is a matrix of speech features and $w^{(i)}$ is the set of labels (word transcription) of speech.
- Unsupervised training data: when one has only $\mathcal{D}_u = \left\{ x^{(i)}_{1:T_i} \right\}_i$
- Semi-supervised training data: when one has only $\mathcal{D}_u = \left\{ x^{(i)}_{1:T_i} \right\}_i$ and $\mathcal{D}_s = \left\{ (x^{(i)}_{1:T_i}, w^{(i)}) \right\}_j$.
- Extra language data $\mathcal{D}_\ell = \left\{ , w^{(i)} \right\}_j$ useful since HMMs are generative models. Acoustic and language data can have little overlap.
- Language data can be either: 1) sequence of words, 2) sequence of words where center time-mark is known, and 3) sequence of words where word boundaries are known.
Supervised Speech Data

- We saw last time the difficulty in collecting quality training data.
- Let’s assume supervised case: \( \mathcal{D} = \left\{ (x_{1:T_i}^{(i)}, w^{(i)}) \right\}_i \), where \( x_{1:T_i}^{(i)} \) is a matrix of speech features and \( w^{(i)} \) is the set of labels (word transcription) of speech.

Speech data - vocabulary size

- In \((x_{1:T_i}^{(i)}, w^{(i)})\), \( x_{1:T_i} \) is a sequence of length \( T_i \) and \( w^{(i)} \) is a sequence also of variable length (length \( N_i \)).
- Vocabulary size (also called lexicon size, or number of types) is key — the bigger the vocabulary, the harder the problem becomes.
- This is an issue of confusability.
- 10 word vocabulary is much less confusable than 250,000 word vocabulary, much more likely that two words are almost the same as vocabulary increases.
Speech data - what is in the lexicon

- \( w^{(i)} = (w_1^{(i)}, w_2^{(i)}, \ldots, w_N^{(i)}) \) is an ordered list of lexical items, but what are they?
- Conversational speech: filled with disfluencies!
- From Wikipedia:

A speech disfluency . . . is any of various breaks, irregularities, or non-lexical vocables that occurs within the flow of otherwise fluent speech. These include false starts, i.e. words and sentences that are cut off mid-utterance, phrases that are restarted or repeated and repeated syllables, fillers i.e. grunts or non-lexical utterances such as “uh”, “erm” and “well”, and repaired utterances, i.e. instances of speakers correcting their own slips of the tongue or mispronunciations (before anyone else gets a chance to).

<table>
<thead>
<tr>
<th>Name</th>
<th>Abbrev.</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repetition or correction</td>
<td>REP</td>
<td>Exact repetition or correction of words previously uttered. A correction may involve substitutions, deletions, or insertions of words. Correction continues with the same idea/train of thought started previously.</td>
<td>If I can’t find don’t know the answer myself, I will find it. If if nobody wants to influence President Clinton</td>
</tr>
<tr>
<td>False Start</td>
<td>FS</td>
<td>An utterance is aborted and restarted with a new idea or train of thought</td>
<td>We’ll never find what about next month?</td>
</tr>
<tr>
<td>Filled Pause</td>
<td>UH</td>
<td>All other filler words without semantic content</td>
<td>yeah, oh, okay etc.</td>
</tr>
<tr>
<td>Interjection</td>
<td>IN</td>
<td>Restricted group of non-lexicalized sounds indicating affirmation or negation. E.g., back-channeling, with no obvious semantic content</td>
<td>uh-huh, mmm, umm, uh-uh</td>
</tr>
</tbody>
</table>
Speech Disfluency Types

http://www.is.cs.cmu.edu/11-733/2003/Slides/FeiHuang.pdf

<table>
<thead>
<tr>
<th>Name</th>
<th>Abbrev.</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editing Term</td>
<td>ET</td>
<td>Phrases that occur between that part of a disfluency which will be corrected and the actual repair. They refer explicitly to the words that just previously have been said indicating that they will be edited.</td>
<td>We need two tickets, I'm sorry three tickets for the flight to Boston.</td>
</tr>
<tr>
<td>Discourse Marker</td>
<td>DM</td>
<td>Words that are related to the structure of the discourse in so far that they help beginning or keeping a turn or serve no acknowledgment. They do not contribute to the semantic content of the dialogue.</td>
<td>Well this is a good idea. This is, you know, a pretty good solution to our problem.</td>
</tr>
</tbody>
</table>

Disfluencies Example

- file://spon2n.wav
- file://radio_lab_disfluencies.wav
Statistics about Disfluencies

Estimations exist of the following (see the work of Liz Schriberg for more details).

- The number of disfluency per sentence grow linearly with sentence length.
- The number of fluent sentences decrease exponentially over the sentence length.
- Disfluencies are not distributed uniformly over positions.
- The frequency of disfluency decreases exponentially w.r.t. its length (number of words).

Speech data — speech annotation

- Repetition/correction/restart/filler: What do these sound like?
- How are they annotated in trained data?
- Annotating speech data (taking $x$ and producing labels $w$) is difficult.
- Time it takes to do this is many times real time.
- There is no universal agreement — word boundaries are ambiguous.
Word boundaries are ambiguous

Spectra of cuttings obtained from Switchboard conversation sw02423, cutting the B-channel from 1:44:741s to 1:45:704s. The arrows show word boundaries hypothesized by trained speech researchers asked to annotate these cuttings.

W₁ = money
W₂ = on
W₃ = him

Word boundaries are ambiguous

Spectra of cuttings obtained from Switchboard conversation sw02423, cutting the A-channel from 8:05:530s to 8:06:622s. The arrows show word boundaries hypothesized by trained speech researchers asked to annotate these cuttings.

W₁ = spent
W₂ = one
W₃ = winter
W₄ = north
Speech data — prosody

- What to do about prosody?
- Suprasegmental: effects extend beyond the phoneme into units of syllables, words, phrases, and sometimes even sentence.
- Prosody: perception of rhythm, intonation, and stress patterns — helps the listener understand the speech message by pointing out important words and by cueing logical breaks in the flow of an utterance.
- Helps mentally process speech,
- Determines message: “Joe has studied”, is it a question or a statement, depends on prosody.
- Dialog acts, reflect the functions that an utterance serves in a discourse (questions, statements, back-channels,)

Dialog acts

Dialog acts (speech acts) group families of surface utterances into functional classes.
- Directive: The speaker wants the listener to do something.
- Commissive: The speaker indicates that s/he herself will do something in future.
- Expressive: The speaker expresses his or her feelings or emotional response.
- Representative: The speaker expresses his or her belief about the truth of a proposition.
- Declarative: Speaker’s utterance causes a change in external, nonlinguistic situation.

Should these be annotated as well?
Rich annotation/Rich Transcription

- Rather than just the words, we might augment lexicon with a wide variety of other aspects of the speech signal.
- Some might occur concurrently with the words, so really $w^{(i)}$ might be a list of discrete vectors (i.e., at any time, we have a word, a certainty about a word boundary, a dialog act, a prosodic category, etc.)
- Above is a “shallow parse” of the speech signal, but higher grammatical structure we might also wish to infer.
- This makes both the human transcriber’s job and the speech recognizers job much harder.

Speech data — text normalization

- One last item is that the same utterances need to be annotated the same way.
- We need to normalize the text so that punctuation, stress marks, indications of disfluency are the same.
- Text normalization: conversion from the variety symbols, numbers, and other nonorthographic entities of text into a common orthographic transcription suitable for subsequent phonetic conversion.
- Normally done on text corpora where end-of-sentence detection, the expansion of abbreviations, and the treatment of acronyms and numbers, is necessary to make the text processable in a unified way.
- Annotator differences, different annotators should ideally do the same thing, need to normalize in the same way.
Supervised Speech Data

We saw last time the difficulty in collecting quality training data.

- Lets assume supervised case: \( D = \left\{ (x_{1:T_i}^{(i)}, w_i^{(i)}) \right\}_i \), where \( x_{1:T_i}^{(i)} \) is a matrix of speech features and \( w_i^{(i)} \) is the set of labels (word transcription) of speech.

HMM: As factorization

Using the conditional independence statements mentioned above, we can derive the following factorization:

\[
p(x_{1:T}, q_{1:T}) = p(x_T, q_T|x_{1:T-1}, q_{1:T-1})p(x_{1:T-1}, q_{1:T-1})
\]

\[
= p(x_T|q_T, x_{1:T-1}, q_{1:T-1})p(q_T|x_{1:T-1}, q_{1:T-1})
\]

\[
p(x_{1:T-1}, q_{1:T-1})
\]

\[
= p(x_T|q_T)p(q_T|q_{T-1})p(x_{1:T-1}, q_{1:T-1})
\]

\[
= \ldots
\]

\[
= p(q_1) \prod_{t=2}^{T} p(q_t|q_{t-1}) \prod_{t=1}^{T} p(x_t|q_t)
\]

This last equation is the classical factorization expression for an HMM joint distribution over \( x_{1:T}, q_{1:T} \).
HMM parameters

- Parameters of HMM, depend on nature of underlying Markov chain.
- If time-homogeneous, we have an initial state distribution (typically \( \pi \)) with \( p(Q_1 = i) = \pi_i \), and a state transition matrix \( A \).
- We also have the set of observation distributions 
  \[ b_j(x) = p(X_t = x | Q_t = j) \]
  in the time-homogeneous case. In time homogeneous case, we might have \( b_{t,j}(x) \). Also, \( B = \{b_i(\cdot)\}_i \).
- HMM, conditional on parameters \( \lambda \), is given as 
  \[ p(x_{1:T} | \lambda) \].
- All three parameters (initial distribution, transition matrix, and observation distribution) together refer to using \( \lambda = (\pi, A, B) \).
- Sampling from an HMM means: 1) first randomly choose an assignment to \( Q_{1:T} \) and then 2) randomly choose an assignment to \( X_{1:T} \).
- Each new \( X \) sample requires a new \( Q \) sample.

Speech Data

- We train on all of the data \( D = \left\{ (x^{(i)}_{1:T_i}, w^{(i)}) \right\}_i \), but notationally it is sometimes easy to assume we have only one utterance sample that is very long.
Parameter Training

- Still two problems to solve.
  - Problem 3: Maximum Likelihood Training
  - Problem 4: Discriminative Training (next week)
  - Why not just do:
    \[
    \frac{\partial}{\partial \lambda} p_{\lambda}(\bar{x}_{1:T} | M) = 0 \tag{12.1}
    \]
    and then solve for \( \lambda \)?
- Recall in an HMM:
  \[
p(x_{1:T}) = \sum_{q_{1:T}} \prod_{t} p(x_t | q_t)p(q_t | q_{t-1}) \tag{12.2}
  \]
- Parameters are “coupled”, no closed form solution since sums do not distribute into products with separate parameters (parameters for each factor are “dependent” on each other).

Could continue with derivatives, and iterate with old to produce new, but does this converge?

EM algorithm helps with this. Missing data consists of hidden variable assignments, take expected full log likelihood conditioned on data and a “previous” guess of the parameters, and then iterate.
To decide which queries to compute, should know which ones we want. If learning HMM parameters with EM, what queries do we need?

- $X_{1:T} = \bar{x}_{1:T}$ observed, $Q_{1:T}$ hidden variables, $\lambda$ are parameters to learn, and $\lambda^p$ are the previous iteration parameters. EM then repeatedly optimizes the following objective:

\[
E_{p(x_{1:T}, q_{1:T} | \lambda^p)} \left[ \log p(x_{1:T}, q_{1:T} | \lambda) \right]
\]

We use an auxiliary function $Q(\lambda, \lambda^g)$ which is a function only of $\lambda$:

\[
f(\lambda) = Q(\lambda, \lambda^g)
\]

\[
= E_p(x_{1:T}, q_{1:T} | \lambda^p) \left[ \log p(x_{1:T}, q_{1:T} | \lambda) \right]
\]

\[
= \sum_{q_{1:T}} p(x_{1:T}, q_{1:T} | \lambda^g) \left[ \log p(x_{1:T}, q_{1:T} | \lambda) \right]
\]

\[
= E_p \left[ \log \prod_t p(q_t | q_{t-1}, \lambda) p(x_t | q_t, \lambda) \right]
\]

\[
= E_p \left[ \sum_t \log p(q_t | q_{t-1}, \lambda) + \sum_t \log p(x_t | q_t, \lambda) \right]
\]

\[
= \left( \sum_t \sum_{ij} p(Q_t = j, Q_{t-1} = i | x_{1:T}, \lambda^p) \log p(Q_t = j | Q_{t-1} = i, \lambda) \right)
\]

\[
\quad + \sum_t \sum_i p(Q_t = i | x_{1:T}, \lambda^p) \log p(x_t | Q_t = i, \lambda) \right) \right)
\]

\[
p(x_{1:T} | \lambda^g)
\]
So this means that for EM learning, we need \( p(Q_t = i|x_{1:T}) \) and \( p(Q_t = j, Q_{t-1} = i|x_{1:T}) \) in an HMM.

Note that we already know how to compute them with the forward/backward computations.

Lets break it into the three parameters, initial distribution, transition matrix, and observations:

\[
Q(\lambda, \lambda^g) = \sum_{q_1:T} \left[ \log p_\lambda(q_1) + \sum_{t=1}^{T} \log p_\lambda(x_t|q_t) \right. \\
\left. + \sum_{t=2}^{T} \log p_\lambda(q_t|q_{t-1}) \right] p(x_{1:T}, q_{1:T}|\lambda^g) 
\]

(12.10)

(12.11)

In expected value expression, parameters have again decoupled.

Goal: find optimal \( \lambda \) for given fixed \( \lambda^g \).

Have three terms in sum for each of three parts of \( \lambda \), each of the three parts are parameter “separate”, each is possible and easier to solve independently.

note: for posterior weighting using previous “guessed” parameters, we have option to either use:

\[
p(x_{1:T}, q_{1:T}|\lambda^g) 
\]

(12.12)

or

\[
p(q_{1:T}|x_{1:T}, \lambda^g) 
\]

(12.13)

since other factor \( p(x_{1:T}) \) doesn’t change optimal solution:

Goal:

\[
\lambda^* \in \arg\max_{\lambda} Q(\lambda, \lambda^g) 
\]

(12.14)
EM - initial state, \( p(q_1), \pi \)

- We optimize first term:
  \[
  \sum_{q_1:T} \log p_\lambda(q_1)p(x_{1:T}, q_{1:T} | \lambda^g) = \sum_{q_1} \log p_\lambda(q_1)p(x_{1:T}, q_1 | \lambda^g)
  \]
  \[(12.15)\]

- Need to stay within probability simplex, so use Lagrange multiplier \( \alpha \)
  \[
  \alpha (\sum_{q_1} p_\lambda(q_1) - 1)
  \]
  \[(12.16)\]

- get:
  \[
  \frac{\partial}{\partial \pi_j} \left( \sum_i \log \pi_i p(x_{1:T}, Q_1 = i | \lambda^g) + \alpha (\sum_i \pi_i - 1) \right) = 0
  \]
  \[(12.17)\]

- or
  \[
  \frac{1}{\pi_j} p(x_{1:T}, Q_1 = j | \lambda^g) + \alpha = 0
  \]
  \[(12.18)\]

This gives
\[
\alpha \pi_j = -p(x_{1:T}, Q_1 = j | \lambda^g)
\]
\[(12.19)\]

Then using:
\[
\frac{\partial}{\partial \alpha} \left( \sum_i \log \pi_i p(x_{1:T}, Q_1 = i | \lambda^g) + \alpha (\sum_i \pi_i - 1) \right) = 0
\]
\[(12.20)\]

Gives
\[
\alpha \sum_j \pi_j = - \sum_j p(x_{1:T}, Q_1 = j | \lambda^g)
\]
\[(12.21)\]

Or
\[
\alpha = -p(x_{1:T} | \lambda^g)
\]
\[(12.22)\]
EM - initial state, $p(q_1)$, $\pi$

- This gives the solution for the first term:
  $$\pi_j = \frac{p(x_1:T, Q_1 = j | \lambda^g)}{p(x_1:T | \lambda^g)} = p(Q_1 = j | \lambda^g, x_1:T) \quad (12.23)$$

- And again, we have an easy way to compute this quantity using $\alpha, \beta$ that is:
  $$p(Q_1 = j | \lambda^g, x_1:T) = \frac{\alpha_j(1) \beta_j(1)}{\sum_q \alpha_q(1) \beta_q(1)} \quad (12.24)$$

EM - state-transition matrix $a_{ij}$

- To optimize the state transition matrix $[A]_{ij} = a_{ij}$
  $$\sum_{q_1:T} \left[ \sum_{t=2}^T \log p_{\lambda}(q_t | q_{t-1}) \right] p(x_{1:T}, q_{1:T} | \lambda^g) = \sum_i \sum_j \sum_{t=2}^T \log p_{\lambda}(j|i)p(x_{1:T}, Q_{t-1} = i, Q_t = j | \lambda^g) \quad (12.25)$$

- Using Lagrange multipliers again (for the transition probabilities, one for each row), we get the solution:
  $$a_{ij} = \frac{\sum_{t=2}^T p(x_{1:T}, Q_{t-1} = i, Q_t = j | \lambda^g)}{\sum_{t=2}^T p(x_{1:T}, Q_{t-1} = i | \lambda^g)} \quad (12.27)$$
EM - state-transition matrix $a_{ij}$

- We saw last time that this was easily computable, again from the $\alpha$ and $\beta$ quantities, namely
  \begin{equation}
  \xi_{i,j}(t - 1) = \frac{\beta_t(j) a_{i,j}^g p(x_t | Q_t = j) \alpha_{t-1}(i)}{p(x_1:T | \lambda^g)}
  \end{equation}

- Thus, we get:
  \begin{equation}
  a_{ij} = \frac{\sum_{t=2}^{T} \xi_{ij}(t)}{\sum_{t=2}^{T} \gamma_i(t)} = \text{expected number transitions from } i \text{ to } j \over \text{expected number of times in state } i
  \end{equation}

EM - observation distributions

- From distributed law followed by probability marginalization:
  \begin{equation}
  \sum_{q_1:T} \sum_{t=1}^{T} \log p_\lambda(x_t | q_t) p(x_1:T, q_1:T | \lambda^g)
  \end{equation}

- How to optimize depends on form of observation distribution $p(x|q)$.
- Discrete case:
  \begin{equation}
  p(x_t | q_t) = \prod_{k=1}^{K} \mathbf{1}(x_t = x^{(k,q_t)})
  \end{equation}

  \text{SO}

  \begin{equation}
  \log p(x_t | q_t) = \sum_{k=1}^{K} \mathbf{1}(x_t = x^{(k,q_t)}) \log p_{k,q_t}
  \end{equation}
EM - multinomial observation distributions

- We get:
  \[
  \sum_{t=1}^{T} \sum_{q_t} \sum_{k=1}^{K} \left( \mathbf{1}(x_t = x^{(k,q_t)}) \log p^{(\lambda)}_{k,q_t} \right) p(x_{1:T}; q_t | \lambda^g) \quad (12.34)
  \]

- Still need probability constraints, so again need Lagrange multiplier. We get:
  \[
  \sum_{t=1}^{T} \sum_{q_t} \sum_{k=1}^{K} \left( \mathbf{1}(x_t = x^{(k,q_t)}) \log p^{(\lambda)}_{k,q_t} \right) p(x_{1:T}; q_t | \lambda^g) + \sum_{q} \alpha_q \left( \sum_{k=1}^{K} p^{(\lambda)}_{k,q} - 1 \right) \quad (12.36)
  \]

The form we get is:

\[
\frac{\partial \text{the above}}{\partial p^{(\lambda)}_{k,q}} = \sum_{t=1}^{T} \mathbf{1}(x_t = x^{(i,q)}) \frac{1}{p^{(\lambda)}_{k,q}} p(x_{1:T}, Q_t = q | \lambda^g) + \alpha_q = 0 \quad (12.37)
\]

Solving for \( \alpha_q \), we get:

\[
\alpha_q = -\sum_{t=1}^{T} p(x_{1:T}, Q_t = q | \lambda^g) \quad (12.38)
\]
EM - multinomial observation distributions

- Leading us to the update:

\[
p_{k,q}^{(\lambda)} = \frac{\sum_{t=1}^{T} 1(x_t = x^{(k,q)}) p(x_{1:T}, Q_t = q|\lambda^g)}{\sum_{t=1}^{T} p(x_{1:T}, Q_t = q|\lambda^g)}
\]  

(12.39)

- Intuition: we are just counting the number of observations equal to \(k\) but weighted by the state-\(q\) occupation probability for each \(t\), and then normalized by the total state-\(q\) occupation probability.

For continuous densities (e.g., Gaussians) derivation is similar.

- For a multivariate Gaussian for each state, we have a mean vector \(\mu_q\) and covariance matrix \(C_q\) for each state.

- Updates take the form:

\[
\mu_q = \frac{\sum_{t=1}^{T} x_t p(x_{1:T}, Q_t = q|\lambda^g)}{\sum_{t=1}^{T} p(x_{1:T}, Q_t = q|\lambda^g)} = \frac{\sum_{t=1}^{T} x_t p(Q_t = q|x_{1:T}, \lambda^g)}{\sum_{t=1}^{T} p(Q_t = q|x_{1:T}, \lambda^g)}
\]

(12.40)

and

\[
C_q = \frac{\sum_{t=1}^{T} (x_t - \mu_q)(x_t - \mu_q)^T p(x_{1:T}, Q_t = q|\lambda^g)}{\sum_{t=1}^{T} p(x_{1:T}, Q_t = q|\lambda^g)}
\]

(12.41)

- So just weighted mean of vectors, or weighted mean of outer-products.

- Q: how to do this in one pass through the data per epoch?
EM iterations

**Algorithm 1:** EM algorithm, top-level iteration

| input | : Training data, auxiliary function $Q(\cdot, \cdot)$ |
| input | : Initial guess at the parameters $\lambda^0$ |

1. $i \leftarrow 0$
2. repeat
3. $i \leftarrow i + 1$
4. $\lambda^i \in \text{argmax}_\lambda Q(\lambda, \lambda^{i-1})$
5. until $\log \left[ \frac{p(\bar{x}_1:T|\lambda^i)}{p(\bar{x}_1:T|\lambda^{i-1})} \right] \leq \tau$ (data log likelihood difference falls below threshold)

- Each iteration is often called an epoch
- Goal is to maximize likelihood, i.e.:
  $$\lambda^* \in \text{argmax}_\lambda \log p(x_1:T|\lambda) = \text{argmax}_\lambda L(\lambda) \quad (12.42)$$
- Why does above iterative approach work?

---

**Theorem 12.4.1**

If $Q(\lambda, \lambda^g) \geq Q(\lambda^g, \lambda^g)$ then $L(\lambda) \geq L(\lambda^g)$ where

$L(\lambda) = p(x_1:T|\lambda) = \sum_{q_1:T} p(x_1:T, q_1:T|\lambda)$ and

$Q(\lambda, \lambda^g) = \sum_{q_1:T} p(q_1:T|\lambda^g) \log p(x_1:T, q_1:T|\lambda)$

**Proof.**

$$
\log \left[ \frac{L(\lambda)}{L(\lambda^g)} \right] = \log \left[ \frac{\sum_{q_1:T} p(x_1:T, q_1:T|\lambda)}{L(\lambda^g)} \right] \quad (12.43)
= \log \left[ \sum_{q_1:T} \frac{p(q_1:T|\lambda^g, x_1:T) p(x_1:T, q_1:T|\lambda)}{p(q_1:T|\lambda^g, x_1:T) L(\lambda^g)} \right] \quad (12.44)
\geq \sum_{q_1:T} p(q_1:T|\lambda^g, x_1:T) \log \frac{p(x_1:T, q_1:T|\lambda)}{p(x_1:T, q_1:T|\lambda^g)} \quad (12.45)
= Q(\lambda, \lambda^g) - Q(\lambda^g, \lambda^g) \geq 0 \quad (12.46)
$$
EM isn’t the only way to learn parameters.

Suppose we wanted to use a gradient descent like algorithm on:

\[ f(\lambda) = \log p(x_{1:T} | \lambda), \]

as in

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

\[
\sum_{q_{1:T}} p(x_{1:T}, q_{1:T} | \lambda) = \frac{\partial}{\partial \lambda} \sum_{q_{1:T}} p(x_{1:T}, q_{1:T} | \lambda)
\]

\[
\frac{1}{p(x_{1:T} | \lambda)}
\]

(12.47)

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

\[
\text{numerator} = \frac{\partial}{\partial a_{ij}} \sum_{q_{1:T}} p(x_{1:T}, q_{1:T} | \lambda) = \frac{\partial}{\partial a_{ij}} \sum_{q_{1:T}} \prod_{t} p(x_{t}|q_{t}) p(q_{t}|q_{t-1})
\]

(12.49)

Define \( \mathcal{T}_{ij}(q_{1:T}) \equiv \{ t : q_{t-1} = i, q_{t} = j \} \) in the following:

\[
\text{numerator} = \frac{\partial}{\partial a_{ij}} \sum_{q_{1:T}} \prod_{t \in \mathcal{T}_{ij}(q_{1:T})} p(x_{t}|q_{t}) \prod_{t \notin \mathcal{T}_{ij}(q_{1:T})} a_{ij} \prod_{t \notin \mathcal{T}_{ij}(q_{1:T})} p(q_{t}|q_{t-1})
\]

(12.50)
HMM - learning with gradient descent

We get

\[
\text{num} = \sum_{q_1:T} \prod_t p(x_t|q_t) \frac{\partial}{\partial a_{ij}} T_{ij}(q_1:T) \prod_{t \notin T_{ij}(q_1:T)} p(q_t|q_{t-1}) \\
= \sum_{q_1:T} \prod_t p(x_t|q_t) \left| T_{ij}(q_1:T) \right| a_{ij}^{-1} \prod_{t \notin T_{ij}(q_1:T)} p(q_t|q_{t-1}) \\
= \sum_{q_1:T} \prod_t p(x_t|q_t)p(q_t|q_{t-1}) \left| T_{ij}(q_1:T) \right| a_{ij} = \sum_{q_1:T} p(x_1:T, q_1:T) \left| T_{ij}(q_1:T) \right| a_{ij} \\
= \frac{1}{a_{ij}} \sum_{q_1:T} p(x_1:T, q_1:T) \sum_t 1\{q_{t-1} = i, q_t = j\} \\
= \frac{1}{a_{ij}} \sum_t \sum_{q_1:T} p(x_1:T, q_1:T) 1\{q_{t-1} = i, q_t = j\} \\
= \frac{1}{a_{ij}} \sum_t p(x_1:T, q_{t-1} = i, q_t = j) \\
\]

This means that, like in EM, for gradient descent learning, we also need for all 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. So clique posteriors are fundamental, we must have a procedure that produces them quickly.
- DTW (and DP) is one early method people used to recognize speech, and is based on templates.