Engineering Tripos Part IIB FOURTH YEAR

MODULE 4F11: SPEECH PROCESSING

Solutions to Examples Paper 1

1. From the diagram, there are about 338 samples for five oscillations, so the fundamental frequency is:

$$16000\frac{5}{338} = 236\text{Hz}$$

There are three cycles of the response within the fundamental period. The most prominent formant is about three times the fundamental frequency, or about 700 Hz.

2. Major Points:
The linear prediction spectrum is the frequency response of the LP all-pole filter. It therefore reflects the inbuilt assumptions of LP analysis i.e. *all-pole model* (lossless, no side-branch acoustic tube), that model parameters are suitably *estimated*, and model *order* selected correctly, excitation has a flat spectrum. LP analysis always provides a smooth spectral representation which will be accurate if the assumptions are satisfied. For autocorrelation LP analysis approximately the same number of speech samples and windowing need to be used as for DFT.
The DFT estimate measures the spectrum directly. Since the the vocal tract excitation causes noise in the spectrum, we can try and estimate vocal tract frequency response by using either direct smoothing of the DFT itself or by using a cepstral smoothing process (take low order cepstral coefficients and then take DFT of those). Note that these smoothing operations (e.g. the number of cepstral coefs to retain) implicitly will have assumptions about the allowed underlying spectral variations.

3. The table shows the working of the HMM. The first lines show the observation sequence and the output probability density from each emitting state.
The forward probability computation is then shown. The probability at the entry state at time 0 is 1.0 and the exit state probability is a_{34} times $\alpha_3(7)$. Note that this is $p(\boldsymbol{O}|\lambda)$.
The backward probabilities are given. Each of the main entries including the entry state is computed using the backward recursion.
The value of $L_j(t)$ is then given as the $\alpha_j(t)\beta_j(t)/p(\boldsymbol{O}|\lambda)$. It can be seen that this sums to one over the emitting states for each observation time.
The Viterbi probabilities are shown. The calculation is identical to the forward probabilities except that the sums are replaced by max operations. The most likely path probability is given in the end state at the final time.
The traceback. Each entry records the best previous state i.e. the state which gave the max in calculating the Viterbi probabilities. The most likely state sequence is obtained by tracing back from the final state.

time t	0	1	2	3	4	5	6	7
o_t		0.2	0.1	0.1	0.5	0.6	0.8	0.7
$b_2(o_t)$		0.5421	0.5586	0.5586	0.4394	0.3936	0.2975	0.3456
$b_3(o_t)$		0.2975	0.251	0.251	0.4394	0.40808	0.5421	0.5156
$\alpha_j(t)$								
state 1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
state 2	0.0	3.794E-01	1.272E-01	4.262E-02	1.124E-02	2.654E-03	4.737E-04	0.0
state 3	0.0	8.925E-02	5.601E-02	2.401E-02	1.593E-02	8.289E-03	4.17E-03	1.818E-03
state 4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.6535E-04
$\beta_j(t)$								
state 1	3.6535E-04	0.0	0.0	0.0	0.0	0.0	0.0	0.0
state 2	0.0	9.122E-04	2.431E-03	5.805E-03	1.284E-02	2.525E-02	4.125E-02	0.0
state 3	0.0	1.95E-04	9.712E-04	4.837E-03	1.376E-02	3.578E-02	8.25E-02	0.2
state 4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
$L_j(t)$								
state 2		0.9521	0.8504	0.6805	0.3970	0.1843	0.0537	0.0
state 3		0.0479	0.1496	0.3195	0.6030	0.8157	0.9463	1.0
Viterbi								
state 1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
state 2	0.0	3.794E-01	1.272E-01	4.262E-02	1.124E-02	2.654E-03	4.737E-04	0.0
state 3	0.0	8.925E-02	3.809E-02	1.277E-02	7.419E-03	2.881E-03	1.249E-03	5.514E-04
state 4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.031E-04
Traceback								
(state at $t-1$)								
state 2		1	2	2	2	2	2	2
state 3		1	2	2	2	3	3	3

4. By definition

$$L_j(t) = p(x(t) = j|\boldsymbol{O}, \lambda)$$

The required equation follows directly from the definition of conditional probability.

The calculation of $L_j(t)$ for the example was given above in the solution for Q3. Note that $L_j(t)$ is heavily biased towards state 2 initially and state 3 finally but in the centre of the sequence the probabilities of being in state 2 or 3 are similar. This is in contrast to the Viterbi case where it flips from state 2 to state 3 at time 4.

The mean estimates for the first iteration follow directly as (b) $\mu_2 = 0.133$ and $\mu_3 = 0.65$ and (b) $\mu_2 = 0.223$ and $\mu_3 = 0.594$ (c.f. the original values of 0 and 1).

5. $P(\tau)$ is just the probability of taking the self transition $\tau - 1$ times followed by a single transition out of the state, i.e.

$$P(\tau) = a^{\tau-1}(1-a)$$

Taking the expectation

$$\begin{aligned}
\mathcal{E}[\tau] &= \sum_{\tau=1}^{\infty} P(\tau)\tau \\
&= \sum_{\tau=1}^{\infty} a^{\tau-1}(1-a)\tau \\
&= (1-a)(1+2a+3a^2+\ldots \\
&= 1+a+a^2+a^3+\ldots \\
&= \frac{1}{1-a}
\end{aligned}$$

Since $a < 1$, $P(\tau)$ is a decaying exponential function. This is clearly not a a good model of speech duration and this is regarded as a major weakness of HMMs.

6.

(a) Viterbi algorithm:
The initialisation condition specifies that only the first state is valid at the start:

$$\phi_i(0) = \begin{cases} 1 & i = 1 \\ 0 & \text{otherwise} \end{cases}$$

```
for t = 1 to T
    for j = 2 to N-1
        $\phi_j(t) = \max_i \left(\phi_i(t-1)a_{ij}\right) b_j(o(t))$
```

where

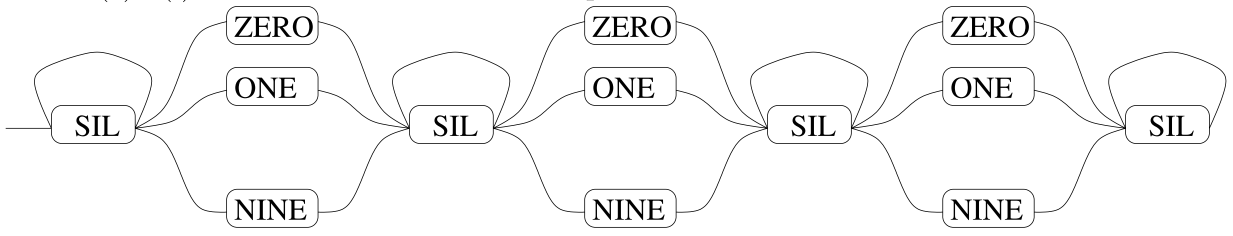
$\phi_j(t)$	is the likelihood to state j at time t while following the most likely path
a_{ij}	is the transition probability from state i to state j
$b_j(o(t))$	is the output probability density of observation $o(t)$ in state j

(b) Aim is to find the most likely path through the HMM. For connected word recognition the HMM is a *sentence* HMM. The Viterbi algorithm is applied to the complete HMM, storing suitable path information. This path information can be stored at the word level (assuming that time points between HMM-internal state transitions are not required). Hence at each time as the path is extended a pointer is preserved as to where that path entered the current model.

The algorithm can be conveniently thought of as operating in two phases. Initially the equations in part (a) are applied internally within each word HMM (assuming for a moment there is no further internal structure) and the values of $\phi_j(t)$ found for all the HMM final states. Then, the values of $\phi_j(t)$ need to be updated for each between word transition and path information stored that contains the value of $\phi_j(t)$, the identity of the emitting word (network node), and a pointer to the path information for the previous words.

At the end of the utterance a particular (best) path has been extended to a network exit node, and can recover the actual sequence of words for that path in reverse order.

(c) (i) SIL between words allows optional silences.



(ii) with no beam search the main components are:

output probabilities	O
word internal path extension	I
word external path extension	E

For loop model, total is $O+I+E$, for the linear model the total is $O+3I+3E$. However, O may dominate the computation, in which case all are similar.

With beam search, probably similar gains for each model but with a larger beam for the linear version.

7. (a) recognition units must be:

- small in number so they can be estimated
- naturally part of a word
- account for speech variability
- well defined and easily identifiable
- extendible to unseen words

Phones are good candidates except that there acoustic realisation is highly context dependent on especially neighbouring phones. Context dependent phone units are a way to address this problem. However, too many triphones are possible - hence the need to have tied parameters.

(b)
- backing off: Define some count for which the model is reasonably well estimated - if not enough instances in training data then use a left context or right context biphone model - if still not enough then use monophone.
- parameter tying: Cluster data into groups which are expected to be from the same/similar distribution and make each group large enough to get a good parameter estimate (for the number of parameters that are to be estimated from the pooled set of contexts). A common way to perform tying is to use decision-tree state clustering in which phonetic contexts are put into equivalence classes by automatically growing a tree structure which divides all e.g. triphones into different groups. The tree is grown by at each step choosing a question (which divides the contexts) to maximise an approximate meansure of the increase in training set likelihood (using single Gaussian assumptions for all distributions). The set of questions are predefined to conisder sets of contexts which would be expected to vary in the same way based on lingustic knowledge, and so can generalise to unseen contexts.

8. (a) A bigram grammar estimates the probability of the next word given a history of one word $\Pr(w(n)|w(n-1))$ while the trigram grammar uses a history of two words, $\Pr(w(n)|w(n-2), w(n-1))$.

Provided the parameters are reasonably well estimated, a trigram grammar has lower perplexity, and therefore generally results in lower word error rates.

(b) Estimate by counting

However need to deal with case of zero counts/unreliable counts. Use discounting to free up some of the probability mass of high counts for zero counts. Use backoff to decide how to distribute this mass over the count zero cases.

(c) Have a unigram table that points to non-backed off bigrams that in turn points to non-backed off trigrams. Can't use a big 3D table as it is too big and hence have to a small amount of effort at run-time to determine if e.g. a particular trigram exists or need to go down the back-off chain.

9. (a)
- non-linear spacing: weight according to information content/resolution of human ear
- cepstral transform: better match feature distribution to diagonal covariance assumption

Neither require extra training data needed - simple and effective, don't really change training time or run time computation (but note that better performance makes things run faster - fewer models active in beam search).

(b)
- Use context dependent models with Gaussian mixture output distributions to model context for co-articulation - and a much more training data

even with back-off/parameter tying. In practice use parameter tying (and decision-tree state tying is now very widely used) to match the number of parameters estimated to the training data available. Note that context-depdendent models will increase the complexity of the search (especially using cross-word context-dependent models). However there is a complex relationship (due to beam-pruning) between actual run-time and complexity here. Normally we need to have contenxt-dependent models to improve accuracy.

- Mixture models: Works in practice because densities are not Gaussian with independent elements. typically 2-32 mixture components per state.

(c)

- 1000 words: Fine if the user knows which 1000 words are in use and doesn't go outside this - otherwise prefer a much larger vocabulary e.g. 60,000 words for general English. Increasing vocabulary slows down recognition, not maximum-likleihood training.
- single pronunciation: Fine for most words, perhaps add a few variants for common words - need to accommodate this into training/recognition but no major changes in training data/computational requirements.

(d)

- Viterbi decoder: Add beam search since this is always useful and use e.g. a tree-structured arrangement of the lexicon to share computation.
- Language model: Huge improvement expected if language model constraints can be used - perhaps perplexity 1000 (or 60k) is reduced to 100 or so. Trigram language models increase search space - relies on effective pruning in the decoder. Need data to estimate the language model for which a large pool of text data is required - acoustic training does not change.

P.C. Woodland, February 2014