Module 4F11 Exam 2005 - Speech Processing
Answers

Question 1

(a) The p^{th} order linear predictor is $\hat{s}(n) = a_1 s(n-1) + \cdots + a_p s(n-p)$. The prediction error sequence is

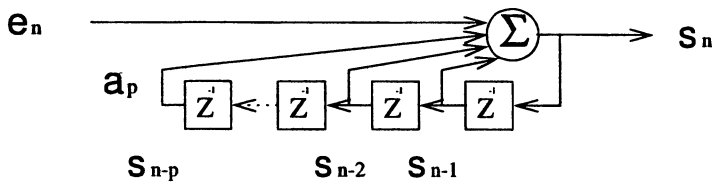
$$e(n) = s(n) - \hat{s}(n) = s(n) - a_1 s(n-1) - \cdots - a_p s(n-p)$$

[10%]

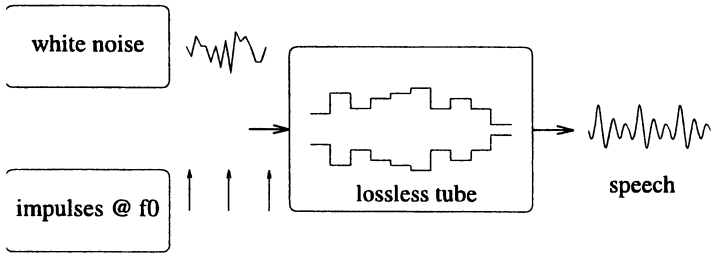
(b) Taking the z-transform of both sides of the first equation, the following results

$$\begin{aligned} e(n) &= s(n) - \textstyle\sum_{i=1}^{p} a_i s(n-i) \\ E(z) &= [1 - \textstyle\sum_{i=1}^{p} a_i z^{-1}] S(z) \\ S(z) &= \frac{1}{1 - \sum_{i=1}^{p} a_i z^{-1}} E(z) \end{aligned}$$

The linear filter is as follows [20%]



(c)



The prediction error sequence can be thought of as excitation to an all-pole model of the vocal tract. The excitation is either quasi-periodic or 'white noise', for voiced or unvoiced speech, respectively. [10%]

(d) The all-pole model requires at least two coefficients for each resonant frequency, so p should be at least 8. [10%]

(e) The polynomial $A(z)$ can be evaluated on the unit circle in the Z plane to find the value of $A(e^{j\omega})$ at equally spaced angular frequencies. The LP spectrum $1/A(e^{j\omega})$ can be found directly. [10%]

(f) Formants can be found directly from the LP spectrum. A simple peak picking algorithm could be used; alternatively, it is possible to look for frequencies at which the second derivative is most negative.

Formants can be found from the LP polynomial by factoring $A(z)$

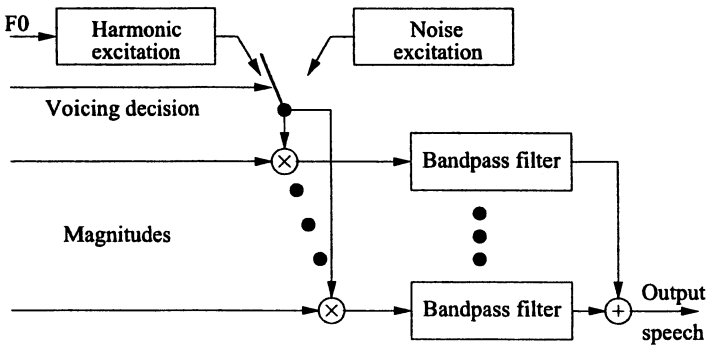
$$\frac{1}{a - \sum_{k=1}^{p} a_k z^{-k}} = \prod_{k=1}^{p} \frac{1}{(1 - z_k z^{-1})}$$

The roots z_k will occur in complex conjugate pairs, and $p/2$ formants can be found selected based on their angular frequency. [25%]

(g) LPC analysis is constrained relative to DFT analysis. LPC will find at most $p/2$ formants that arise from fitting the LP spectrum to the speech power spectrum. Extracting formants from a DFT-based spectrum, via peak-picking, is not constrained by any model of the speech signal, so there is no limit on the number of formants extracted (up to the DFT size), and the DFT-based spectrum is not derived from an all-pole model. [15%]

Question 2

(a)(i) [15%]



(a)(ii) The channel vocoder is based on the source filter models. The fundamental frequency F0 is detected, together with a decision whether the sound is voiced or unvoiced. The vocal tract response is represented in estimates of the magnitude of the spectrum in a number of filter bands. The filter bands are narrow enough that decimation is used to downsample the output of each filter. [15%]

The speech signal is resynthesised by switching the excitation signal between a periodic pulse and white noise, and subsequent band-selective weighting.

(a)(iii) A non-linear filterbank, with filters evenly spaced along the Mel scale could be used so that more bands are devoted to lower frequencies at which human perception is more sensitive. Another perceptual property is auditory masking, when a tone is perceived as diminished in intensity in the presence of a

stronger tone at a close frequency. The analysis stage could be modified so that fewer bits are devoted to a sub-band if the signal in a neighboring sub-band is much louder. [20%]

(b) The goal is to minimise the error signal energy:

$$\epsilon(k) = \sum_{n=0}^{N-1} (s(n) - u(k)h(n-k))^2$$

where $h(n)$ is the impulse response associated with $H(z) = \frac{S(z)}{E(z)}$. By setting the derivative of $\epsilon(k)$ to 0

$$\begin{aligned} \frac{\partial \epsilon(k)}{u(k)} &= \sum_{n=0}^{N-1} (-2\bar{s}(n)h_\gamma(n-k) + 2u(k)h_\gamma^2(n-k)) \\ &= 0 \end{aligned}$$

we get [35%]

$$u(k) \sum_{n=0}^{N-1} h^2(n-k) = \sum_{n=0}^{N-1} s(n)h(n-k)$$

and thus a normalised cross-correlation estimate between $s(n)$ and $h(n)$

$$u(k) = \frac{\hat{r}_{sh}(k)}{\hat{r}_{hh}(0)}$$

Substituting this expression for $u(k)$ into the expression for $\epsilon(k)$ gives the value of the minimized error signal energy.

(c) If the error signal itself was available, the speech signal could be reconstructed without loss. With this as motivation, both techniques attempt to improve the quality of the synthesized speech by optimizing the excitation. CELP uses a codebook to vector quantize the LP residual as part of the encoding processing. The codebook is built on actual residuals obtained from many speech samples. Multipulse excitation attempts to produce an optimized excitation by synthesizing an excitation with the goal of minimizing the mean square distortion of the reconstructed signal. [15%]

Question 3

(a) There are several approaches to parameter initialisation. In a **flat start**, all parameters are set to be the same. For example, a global mean and variance can be estimated over all the training data and used to initialise Gaussian observation distributions. If **hand-segmented data** is available, individual frames of speech can be assigned to estimate state (or subword model) parameters. In some cases, **previous models** trained on another, not too dissimilar, data-set may be available. These can be borrowed and then refined using the available training data. [10%]

(b)(i) The partial path probability $\Phi_j(t)$ can be calculated using a recursion, as follows

$$\Phi_j(t) = \max_i \{\Phi_j(t-1)a_{i,j}b_j(\mathbf{y}_t)\}$$

Given that the model is a left-to-right HMM, appropriate initialization is $\Phi_j(0) = 1$ if $j = 0$ and 0 otherwise. [15%]

(b)(ii) At each time step in the Viterbi algorithm, the identity of the best predecessor state is maintained. This makes it possible to search backwards in time through the trellis starting from state N at time T. [10%]

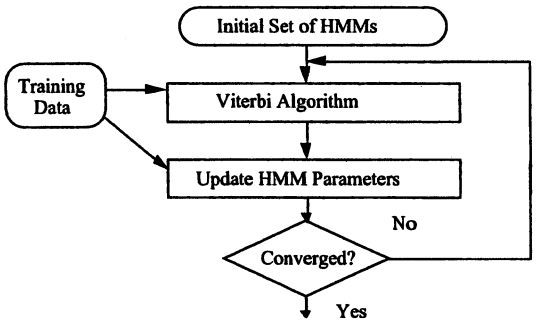
(b)(iii) Since the model is a left-to-right HMM, it is straightforward to determine at what frame when each state in the most likely state sequence begins to generate data. Let t_j denote the frame number at which

state j starts to generate data. This time sequence $\{t_j\}$ defines the *Viterbi Segmentation*, and given this segmentation, the parameters of state s_j can be estimated as [10%]

$$\hat{\mu}_j = \frac{1}{t_{j+1} - t_j} \sum_{t=t_j}^{t_{j+1}-1} y_t$$

$$\hat{\Sigma}_j = \frac{1}{t_{j+1} - t_j} \sum_{t=t_j}^{t_{j+1}-1} [(y_t - \hat{\mu}_j)(y_t - \hat{\mu}_j)']$$

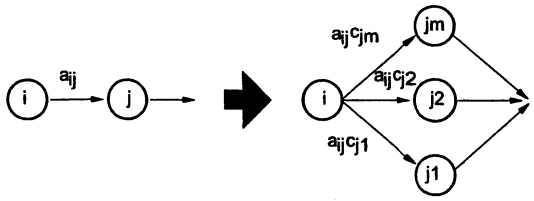
(b)(iv)



[15%]

The likelihood is guaranteed to increase at each iteration. Using $\Phi_N(T)$ and $\hat{\Phi}_N(T)$ to denote the Viterbi likelihood before and after a training iteration, the training algorithm can be halted if the increase in likelihood $\hat{\Phi}_N(T) - \Phi_N(T)$ falls below a threshold. A second possibility would be to monitor the convergence of the model parameters themselves.

(c)(i) To train the multiple Gaussian component system, each component is treated as a separate state. Each of the original emitting states is split into m states, and the mixture weight is absorbed into the transition probability to each of the newly created states [20%]



The Viterbi Segmentation of the training data is found with respect to this model, and the Gaussian parameters are estimated based on the state alignment, e.g. the parameters of the first Gaussian are estimated based on the frames aligned to state $j1$.

The mixture weights are estimated based on the relative frequencies of the component states within the alignment; e.g. if k_{jk} is the number of times that state jm occurs, then $\hat{c}_{jm} = \frac{k_{jk}}{\sum_{k=1}^{m} k_{jk}}$.

(c)(ii) A simple initialisation scheme makes use of the Gaussian parameters estimated for each state in the initial application of Viterbi training. The mixture weights can be set unifomly to $1/m$; the Gaussian covariances can be set to the initial covariance estimates; and the Gaussian means can be initialised as random perturbations of the initial mean estimates. The random perturbation applied to each element of the mean vector can be determined by the initial covariance matrix. [10%]

Question 4

(a) (i) Triphones are models of individual phones that take into account the immediately preceeding and following phonetic context. They can therefore handle within-word and (for cross-word triphones) cross-word co-articulation. Since they are phone models they can be smoothed or tied with other phone contexts. [10%]

(a) (ii) Main issues are robust estimation, but also use of triphone models in decoding. Robust estimation can be solved by smoothing or tying. Furthermore need to be able to decide what to do for triphones that do not occur in training but are needed for test. [10%]

(b) (i) Phonetic decision trees. From notes. A tree grown at the state level (for each of three states left to right models normally used). [30%]

Simple binary decision tree, i.e. yes/no at each node. There are questions about groups of phonetic classes at each node. The trees are automatically constructed by choosing at each stage the question and node that gives the largest increase in training data log likelihood from making the split. Stopping is when there isn't any node with a large enough gain in log likelihood and/or by a occupancy threshold. The tree divides all contexts into equivalence classes and the leaf nodes represent these (for all triphones including those that don't occur in training). An HMM output distribution is built for each clustered state and the data from all contexts shared. The advantages are that there is no need to back-off for unseen triphones; and allows expert knowledge to be incorporated; and allows any degree of context dependency to be simply incorporated. Disadvantages are that requires expert knowledge to specify question set (but not chosen questions!); tree building uses locally optimal decisions.

(b) (ii) Agglomerative bottom up clustering. From notes. Assume that all contexts are distinct and simple models of all training data contexts can be constructed. However to ensure reliable estimates sharing is required. The basic procedure is to first build models for all observed contexts. Then merge similar states using a distance measure between distributions and continue to do this until there is enough data for all states to build robust estimates. This is simple but is unreliable for contexts that occur rarely in training data. It is also unable (without using back-off) to cluster contexts not seen in training data. [20%]

(c) (i) This is the standard way that decision tree clustering is performed, but the problem here isn't covered in the notes. [20%]

So the problem asked for is to estimate the value of Σ and N at any node. To estimate Σ need

$$\mathcal{E}[\mathbf{o}^2] - \mathcal{E}[\mathbf{o}]^2$$

for the data associated with the node. If there are a set of contexts C associated with the node and we have access to the means and variances for each context and conunts then the required value is

$$\sum_{i \in C} N_i \left[\Sigma_i + \mu_i \mu_i'\right] - \sum_{i \in C} N_i \mu_i \mu_i'$$

The main assumption here is that during the clustering and merging statistics that the values of the state/frame alignments don't change so that the original statistics can be combined as above.

(c) (ii) If use mixture distributions we cannot exactly combine the models and so the statistics and the likelihood change is only approximate (unless go back to original data which is very computationally expensive). Since the decision tree process is only locally optimal the increase in computation required is normally viewed as not worthwhile. [10%]

Question 5

(a) The perplexity (PP) is related to the entropy (H) of the language model PP $= 2^H$. The aim is to compute this from the language model. The probability of a word sequence is commonly decomposed into the product of conditional probabilities: [20%]

$$P(w(1)w(2)\dots w(N)) = \prod_{k=1}^{N} P\left(w(k)|w(1)\dots w(k-1)\right)$$

The entropy of this word sequence is given by (letting $N \to \infty$ to obtain a good estimate). However, in practice we will only have S sequences of words to estimate the entropy of the language model on. The entropy is

$$H = -\frac{1}{\sum_{s\in S} N_s}\left(\sum_{s\in S}\sum_{k=1}^{N_s} \log_2 P(w(k)|w(1)\dots w(k-1))\right)$$

When we compute the perplexity using this over a corpus of test sentences to get the Test Set Perplexity.

(b) The probability of a word sequence may be expressed as [30%]

$$P(w(0)w(1)\dots w(N+1)) = \prod_{k=0}^{N+1} P\left(w(k)|w(1)\dots w(k-1)\right)$$

The language model is required to give an estimate of

$$P\left(w(k)|w(1)\dots w(k-1)\right)$$

for any word sequence. For an N-gram restrict the size of the history to the previous $N-1$ words. Thus

$$P(w(k)|w(1)\dots w(k-1)) \approx P(w(k)|w(k-N+1)\dots w(k-1))$$

Basic estimation uses relative frequencies to estimate probabilities. Therefore to estimate the probability of a particular trigram it is necessary to find the "count" (frequency of occurrence) of triple $w_iw_jw_k$ in the training data and then

$$\hat{P}(w_k|w_i,w_j) = \frac{f(w_i,w_j,w_k)}{\sum_{k=1}^{V} f(w_i,w_j,w_k)} = \frac{f(w_i,w_j,w_k)}{f(w_i,w_j)}$$

where $f(a,b,c,\dots)$ = number of times that the word sequence (*event*) "a b c ..." occurs in the training data. This relative frequency approach is the ML estimate of the N-gram parameters.

To allow for word sequences not seen in training need to use discounted counts for the N-grams that are seen. Then we use

$$\hat{P}(w_k|w_i,w_j) \quad = \quad d(f(w_i,w_j,w_k))\frac{f(w_i,w_j,w_k)}{f(w_i,w_j)}$$

where $d(r)$ is a *discount coefficient*. For N-grams not seen enough during training back-off to a lower order distribution weighted by a suitable back-off weight.

The full back-off model is then (e.g. for a bigram)

$$\hat{P}(w_j|w_i) = \begin{cases} d(f(w_i,w_j))\frac{f(w_i,w_j)}{f(w_i)} & f(w_i,w_j) > C \\ \alpha(w_i)\hat{P}(w_j) & \text{otherwise} \end{cases}$$

$\alpha(w_i)$ is the *back-off* weight, it is chosen to ensure that

$$\sum_{j=1}^{V} \hat{P}(w_j|w_i) = 1$$

and C is the N-gram cut-off point (i.e. only N-grams that occur more frequently than this are retained in the final model). The value of C also controls the size of the resulting language model.

(c)(i) For the training data, the entropy H can be found from the model itself by summing over the vocabulary to find the expected value of the log probability. [15%]

$$H = -\sum_{v=1}^{V} P(\omega_v) \log_2 P(\omega_v)$$

where ω_v is the **word type** for the vth vocabulary item. [Note this is not covered in the notes]

(c)(ii) Similarly for a bigram the training data H can be found as [15%]

$$H = -\sum_{u=1}^{V} P(\omega_u) \sum_{v=1}^{V} P(\omega_v|\omega_u) \log_2 P(\omega_v|\omega_u)$$

(d) Normally expected that the training set PP will be lower than the test set perplexity, and possibly much lower. For a fixed sized training corpus the larger the number of language model parameters the larger this difference will be. If discounting is not used the test set perplexity may be infinite, and appropriate discouting will increase the training set perplexity and reduce the test set perplexity. [20%]