Number:

Problem	Points	Score
1 (a)	15	
1 (b)	10	
1 (c)	15	
2 (a)	15	
2 (b)	15	
2 (c)	10	
2 (d)	10	
3 (a)	10	
Total	100	

Notes:

1. The exam is closed books and notes. You are allowed one 8 1/2” x 11” double-sided sheet of notes.

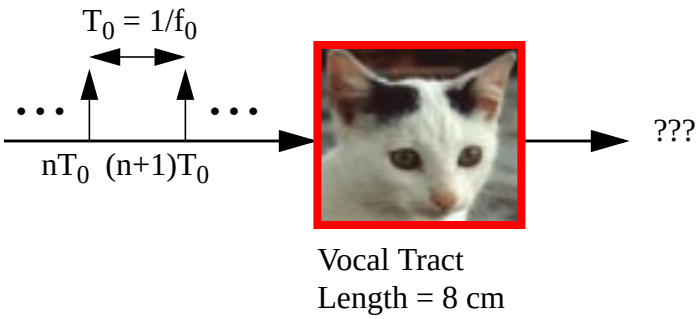
2. Please indicate clearly your answer to the problem by some form of highlighting (underlining).

3. Your solutions must be legible and easy to follow. If I can’t read it or understand it, it is wrong. Random scribbling will not receive credit.

4. Please show ALL work. Answers with no supporting explanations or work will be given no credit.

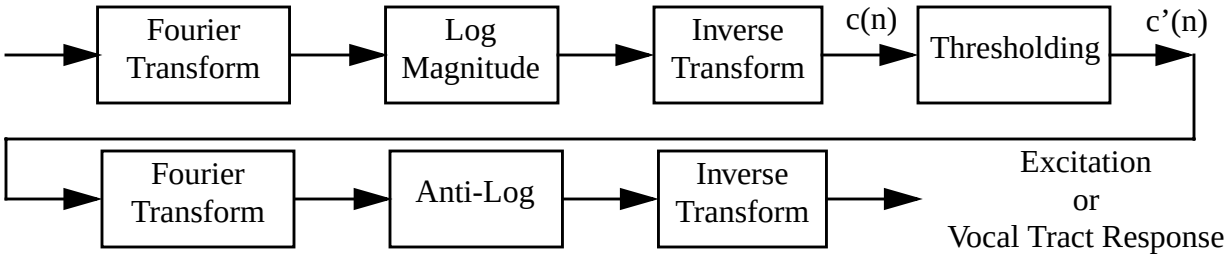
5. Several problems on this exam are fairly open-ended. Since the evaluation of your answers is obviously a subjective process, we will use a market place strategy in determining the grade. Papers will be rank-ordered in terms of the quality of the solutions, and grades distributed accordingly.

1. Consider the system shown to the right — let's call it a "cat synthesizer". Assume the properties of the cat's vocal tract are the same as a human vocal tract, but the length is shorter. Assume a fundamental frequency (f_0) of 100 Hz. Assume the input is a periodic pulse train as shown.



(a) Design a system to separate the excitation signal from the vocal tract frequency response. Be as specific as possible (e.g., numbers, frequency responses, etc.).

We can use a classic homomorphic deconvolution system to solve this problem:



To recover the vocal tract response, the thresholding function simply discards all values of c(n) greater than N samples (to recover the excitation, we do the opposite). N is related to the fundamental frequency of the signal — 1/100 secs in this case. Given that the length of the cat's vocal tract is 8 cm, a sample frequency of 8 kHz and N=40 should be adequate. See the lecture notes for a more detailed explanation of why we rely on the fundamental frequency to determine the value of the threshold, and how we separate the excitation signal.

(b) Explain the inaccuracies of this model if a comparable system were used to process real human speech.

The human speech signal is actually composed of three things: excitation (white noise or impulse train), glottal pulse shaping, and the vocal tract. The above analysis does not necessarily separate the glottal pulse shape from the vocal tract, since the frequency response of the glottal pulse shape will influence the shape of the cepstral signal in the quefrency domain. This is due to the fact that the output signal is a convolution of the pulse train with the glottal pulse shape *and* the vocal tract. Not surprisingly, cepstral techniques tend to accentuate harmonics in the signal — one of the things that has made them unattractive for certain speech processing applications.

(c) Suppose you observe the following features vectors from this system:

$$x[n] = \begin{bmatrix}1\\0\end{bmatrix} \quad \begin{bmatrix}0\\-1\end{bmatrix} \quad \begin{bmatrix}1\\1\end{bmatrix} \quad \begin{bmatrix}-1\\-1\end{bmatrix} \quad \begin{bmatrix}-1\\2\end{bmatrix} \quad \begin{bmatrix}0\\-1\end{bmatrix}$$

Design a prewhitening transformation that decorrelates the data and represents an orthonormal transformation. Demonstrate that this transformation works by processing the above data.

Observe that this is a zero-mean signal ($\sum \bar{x}(n) = 0$. We can compute the covariance:

$$C[i,j] = \sum_{n=0}^{N-1} (x(i)-\mu_i)(x(j)-\mu_j) = \begin{bmatrix}4 & 0\\ 0 & 8\end{bmatrix}$$

Thank goodness it is a diagonal covariance matrix! We can use "variance-weighting":

$$\begin{bmatrix}y_1\\y_2\end{bmatrix} = \begin{bmatrix}\alpha_1 & 0\\ 0 & \alpha_2\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

What should the values of α_1 and α_2 be? Let's try the inverse of the standard deviation:

$$\begin{bmatrix}\alpha_1 & 0\\ 0 & \alpha_2\end{bmatrix} = \begin{bmatrix}\frac{1}{\sigma_1} & 0\\ 0 & \frac{1}{\sigma_2}\end{bmatrix} = \begin{bmatrix}\frac{1}{2} & 0\\ 0 & \frac{1}{2\sqrt{2}}\end{bmatrix}$$

If we filter the above signal by this matrix, we get:

$$\tilde{x}[n] = \begin{bmatrix}1/2\\0\end{bmatrix}\begin{bmatrix}0\\-1/(2\sqrt{2})\end{bmatrix}\begin{bmatrix}1/2\\1/(2\sqrt{2})\end{bmatrix}\begin{bmatrix}-1/2\\-1/(2\sqrt{2})\end{bmatrix}\begin{bmatrix}-1/2\\2/(2\sqrt{2})\end{bmatrix}\begin{bmatrix}0\\(-1)/(2\sqrt{2})\end{bmatrix}$$

Is the signal still zero mean? Let's see:

$$\sum \tilde{\bar{x}}(n) = \begin{bmatrix}0\\0\end{bmatrix}$$

We can compute the covariance of the filtered signal as:

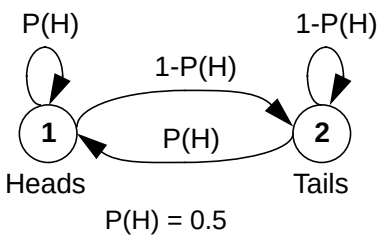
$$\tilde{C}[i,j] = \sum_{n=0}^{N-1} (\tilde{x}(i)-\tilde{\mu}_i)(\tilde{x}(j)-\tilde{\mu}_j) = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

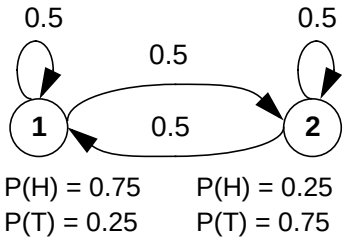
Amazing! Therefore, the transformation:

$$\boldsymbol{T} = \begin{bmatrix}\frac{1}{2} & 0\\ 0 & \frac{1}{2\sqrt{2}}\end{bmatrix}$$

is an orthonormal transformation (e.g., the covariance of the transformed data is an identity matrix).

2. A system ("black box") outputs the sequence "HTT". Consider these two models:





Assume the probability of starting in each state (initial probabilities) are equal.

(a) Apply the principle of dynamic programming to find which model was the most likely model to generate this sequence.

Model 1: P("HTT"|M1) = (0.5)(0.5)(0.5) = 0.125 = 0.5^3

State:	1	2	3
2	N/A	(0.5)(0.5) / S1	(0.5)(0.5)(0.5) / S2
1	0.5	N/A	N/A
Obs:	H	T	T

Model 2: P("HTT"|M2) = 0.02637

State:	1	2	3
2	(0.5)(0.25) 0.125	(0.375)(0.5)(0.75) / S1 0.07031 / S1	(0.07031)(0.5)(0.75) / S2 0.02637 / S2
1	(0.5)(0.75) 0.375	(0.375)(0.5)(0.25) / S1 0.04687 / S1	(0.07031)(0.5)(0.25) / S2 0.00879 / S2
Obs:	H	T	T

Note that after the second symbol, the probabilities are lower than model 1, so model 2 is not possible, and there is no reason to compute t=3 (another time-saving step!).

Hence, P("HTT"|M1) > P("HTT"|M2).

(b) Next, considering all possible state sequences that could have produced this data, find the model that was most likely to have generated this data.

P(M1|"HTT") does not change because it is not a hidden model. Only one path can produce the given output.

Model 2: Let's do this using brute force. There are 8 possible paths. We can compute the probabilities using the spreadsheet shown below:

State Sequ.	$A^2\pi(1)$	Output Probs.				$BA^2\pi(1)$
		H	T	T	B(HTT)	
111	0.125	0.75	0.25	0.25	0.0469	0.0059
112	0.125	0.75	0.25	0.75	0.1406	0.0176
121	0.125	0.75	0.75	0.25	0.1406	0.0176
122	0.125	0.75	0.75	0.75	0.4219	0.0527
211	0.125	0.25	0.25	0.25	0.0156	0.0020
212	0.125	0.25	0.25	0.75	0.0469	0.0059
221	0.125	0.25	0.75	0.25	0.0469	0.0059
222	0.125	0.25	0.75	0.75	0.1406	0.0176
					TOTAL:	0.1250

Now the models are equally probable :)

(c) Assume that "H" represents a value of "0", and "T" represents a value of "1". Also assume the system is capable of outputting continuous-valued outputs (all values in the range [0,1]). Use linear prediction to find the best model of the form $y(n) = ay(n-1) + w(n)$.

Using the autocorrelation method, for a first-order model, $a = R(1)/R(0)$. We can compute these correlation coefficients for $x(n) = \{0, 1, 1\}$

$$R(0) = \sum_{n=0}^{2} x(n)x(n) = 2 \qquad R(1) = \sum_{n=0}^{2} x(n)x(n-1) = (0)(0) + (0)(1) + (1)(1)$$

Hence, $a = \frac{1}{2}$.

If we use the covariance method:

$$a = \frac{C(1,0)}{C(1,1)} = \frac{\sum_{i=0}^{2} x(n-1)x(n)}{\sum_{i=0}^{2} x(n-1)} = \frac{(1)}{(1)} = 1$$

On creative student posed the problem this way:

$$E = \sum_{n=0}^{2} (y(n) - \tilde{y}(n))$$
$$= ((0 - a(0))^2 + (1 - a(0))^2 + (1 - a)^2)$$
$$= (1 - a)^2$$

We can differentiate E w.r.t. a, or just observe that the error is minimum when $a = 1$.

In this problem, we see that the autocorrelation and covariance methods produce different results. Why?

(d) Analyze the differences between these three approaches. Do not simply list their features. Discuss what types of assumptions these models make, and why one might be more appropriate than the other.

The Viterbi and Forward Algorithm, of course, are based on Markov models. Hence, they are significantly different than the linear prediction (LP) model. The LP model assumes the signal is generated from an all-pole filter with a white noise input. It attempts to model the temporal progression of the vectors using a fairly specific function — a weighted sum of previous samples. It is not extremely flexible in its ability to model the data.

The LP model can be viewed as a maximum entropy model of the signal. It makes the least assumptions about the signal outside of the analysis window, but does not require the signal to be zero outside the analysis window. In fact, if we assume the input to the model is Gaussian white noise, we can show that the LP model is a maximum likelihood approach to time series analysis (see Burg's thesis), and that two models can be directly compared using likelihood ratios (Itakura likelihood measure). In this sense, it is somewhat similar to a Markov model based on Gaussian emission probability distributions.

The primary difference between the Viterbi and the Forward algorithm is the assumption made about the state sequences. The Viterbi algorithm assumes we want to model the data by the single-best state sequence. The state sequence, in some sense, becomes observable. The Forward algorithm, on the other hand, allows all possible state sequences. Hence, it is not surprising that the Forward algorithm produces a higher probability for the calculation in 2(b) as compared to the calculation in 2(a) for the second model.

Baum-Welch (BW) training, based on the Forward-Backward algorithm, clearly has the potential for uncovering structure in the data, since the model can self-organize information. The Viterbi algorithm, which is based on principles of dynamic programming, typically trains a model that looks more like a dynamic time-warping approach to speech recognition — the model can be aligned to the input sequence thereby forcing each state in the model to represent specific information (such as the start of a phoneme). The BW approach tends to distribute information across the states in the model.

(I was lenient in grading this problem, but few people provided any real *analysis* of the differences.)

3. Consider a nonlinear model of a feature vector sequence: $y = \begin{bmatrix} a_1 & a_2 \end{bmatrix} \begin{bmatrix} x_1^2 \\ x_2^2 \end{bmatrix} + \begin{bmatrix} b_1 & b_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + c$,

where $\bar{x}$ is the input vector, $\bar{a}$, $\bar{b}$, and c are parameters, and y is an output variable that indicates which of two classes $\bar{x}$ belongs to. Assume y is a probability in the range [0,1] and represents the probability $\bar{x}$ belongs to the first of two classes.

(a) Explain how you might estimate the optimal values of these parameters using a statistical approach covered in the class lectures. Defend the merits of your approach.

Suppose we used a least mean-square error approach to modeling this data (linear regression or linear prediction). What type of equations would we end up with? If you work through the math:

$$E = \sum_n (y - \hat{y})^2 = \sum_n \left(y - \left(\begin{bmatrix} a_1 & a_2 \end{bmatrix} \begin{bmatrix} x_1^2 \\ x_2^2 \end{bmatrix} + \begin{bmatrix} b_1 & b_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + c \right) \right)^2$$

$$\frac{\partial E}{\partial a_i} = \frac{\partial}{\partial a_i} \left[\sum_n \left(y - \left(\begin{bmatrix} a_1 & a_2 \end{bmatrix} \begin{bmatrix} x_1^2 \\ x_2^2 \end{bmatrix} + \begin{bmatrix} b_1 & b_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + c \right) \right)^2 \right] = 0$$

you will realize that you obtain a nonlinear set of equations that must be solved. Years ago, this would have been considered prohibitive. However, with advances in modern computing, you could probably do a decent job of solving these equations.

Now, suppose we set this up as an EM estimation problem. Would the complexity of the resulting equations be any simpler? What would the critical steps be?

A simple approach to estimating these parameters would be to use a gradient descent type approach in which you minimized the error directly by perturbing your guesses for the correct values of the coefficients by an amount proportional to the derivative of the error. Eventually, such a technique would converge... if you avoided getting stuck in local maxima. What approach exploits this type of behavior? Neural networks?

If you think about all the classification algorithms you have at your disposal, would you even want to postulate a form for the model if your ultimate goal was classification? Perhaps you would prefer to let the system learn the best model for the data — particularly if the data was strongly nonlinear...