Computer Science 401 — 8 March, 2010
St. George Campus — University of Toronto

Homework Assignment #3
Due: Thursday, 1 April, 2010 at 12 PM

Speech

TA: Frank Rudzicz

1 Introduction

This assignment introduces you to Matlab, Gaussian mixture modelling, continuous HMMs, and two basic tasks in speech technology: *speaker identification*, in which we try to determine *who* is talking, and *speech recognition*, in which we try to determine *what* was said.

The assignment is divided into two sections. In the first, you will experiment with speaker identification by training mixtures of Gaussians to the acoustic characteristics of individual speakers, and then identify unknown speakers based on these models. In the second section, you will experiment with speech recognition by transcribing data, training hidden Markov models, and analyzing your system's performance with word-error rates.

You will need access to both of the following data sets for each section:

1. **Training**: This contains 30 directories (e.g., `FCJF0/, FDML0/,...`), each representing a unique speaker. Each speaker speaks 9 utterances.
2. **Testing**: This contains 1 directory of 30 utterances, each spoken by a unique but unknown speaker from among the speakers in the training set.

Every utterance is represented by a set of 5 files (i.e., `*.wav, *.mfcc, *.phn, *.txt, *.wrd`); these formats are described in Appendix A.

You will also need the CMU dictionary, which contains the phonetic expansions of some common words.

All of the data can be found on CDF in */u/cs401/speechdata*.

---

2 Speaker Identification

Speaker identification is the task of correctly identifying speaker s_c from among S possible speakers $s_{i=1..S}$ given an input speech sequence X, consisting of a succession of 42-dimensional real vectors. In the interests of efficiency, we'll be doing this over 14-dimensional vectors in this assignment. Each vector represents a small 16ms unit of speech called a *frame*. Speakers are identified by training data that are ascribed to them. This is a discrete classification task (choosing among several speakers) that uses continuous-valued data (the vectors of real numbers) as input.

[0]Copyright © 2009–10, Gerald Penn and Frank Rudzicz. All rights reserved.

2.1 Gaussian Mixture Models

The manner by which spectral features are measured from digitized speech makes independence assumptions about the energies of the different frequencies present in the source signal that are untrue. As a result, the spectra represented by these feature vectors look very unlike the spectra of real speech signals. A common method for dealing with this incongruity is to attempt to model the difference between the measured vectors and those of the "true" signal as noise in a communication channel. This noise, like most other noise in nature, is assumed to be normally distributed.

Gaussian mixture models represent data as the output of a generative process that knows about just a few "true" signals, called *components*, communicated over a noisy channel. GMMs are often used to smoothe over abrupt changes or gaps resulting from sparse data or unwarranted independence assumptions. They can tightly constrain large-dimensional data by using a small number of components, but can, with more components, model other density distributions than Gaussians. Sometimes, they are simply used because the domain being modelled appears to have multiple modes. In the present case, we use them because it is assumed that speakers can be identified by a small number of highly complex spectral patterns. Each component in our GMMs will correspond to one of these patterns.

As mentioned in lecture, a mixture model is much like a game show. First, you pick a door, then you have to play the game behind the chosen door to win. Given M components, GMMs are modelled by a collection of parameters, $\theta = \{\omega_{m=1..M}, \mu_{m=1..M}, \Sigma_{m=1..M}\}$, where ω_m is the probability that component m will be chosen. These are subject to the constraint that $\sum_m \omega_m = 1$. Once a component has been chosen, the "game" is a multivariate Gaussian distribution, which is characterized by that component's mean, μ_m, and covariance matrix, Σ_m. For reasons of computational efficiency, we will reintroduce some independence assumptions by assuming that every component's covariance matrix is diagonal, i.e.:

$$\Sigma_m = \begin{pmatrix} \sigma_m^2[1] & 0 & \cdots & 0 \\ 0 & \sigma_m^2[2] & \cdots & 0 \\ \vdots & & & \\ 0 & 0 & \cdots & \sigma_m^2[d] \end{pmatrix}$$

for some vector $\vec{\sigma}_m^2$. So only $d = 14$ parameters are necessary to characterize a component's (co)variance.

Given these parameters, the probability of generating a particular vector, $\vec{v}$, is:

$$p\left(\vec{v}|\theta_s\right) = \sum_{m=1}^{M} \omega_m b_m(\vec{v}) \tag{1}$$

where b_m is defined in Appendix B.

2.2 Training Gaussian Mixture Models [10 marks]

Your first task will be to train one M-component Gaussian mixture model (GMM) for each of the speakers in the **Training** data set. Specifically, for each speaker s, train the parameters $\theta_s = \{\omega_{m=1..M}, \mu_{m=1..M}, \sigma^2_{m=1..M}\}$ according to the method described in Appendix B. In all cases, assume that covariance matrices σ_m^2 are diagonal. Start with $M = 8$. You'll be asked to experiment with that in Section 2.4.

Your Matlab code should include a function `gmmTrain` which implements Algorithm 1. `gmmTrain` should use Algorithm 1 to train a GMM for every speaker in succession. Note that the parameter ϵ is adjustable. See Appendix B for additional details.

Input: MFCC data X
begin
  `Initialize` θ
  $i := 0$
  $prev\_L = -\infty$
  repeat
    $L =$ `ComputeLikelihood` (X, θ)
    $\theta =$ `UpdateParameters` (θ, X, L)
    $improvement = L - prev\_L$
    $prev\_L = L$
    $i++$
  until $i =< MAX\_ITER$ **and** $improvement >= \epsilon$
end

Algorithm 1: GMM training algorithm.

2.3 Classification with Gaussian Mixture Models [10 marks]

Given a set of trained parameters θ_s for speaker s, assume that the log likelihood of a test sequence $\tilde{X}$ being uttered by that speaker is:

$$\log P\left(\tilde{X}|\theta_s\right) = \sum_{t=1}^{T} \log p\left(\vec{x}_t|\theta_s\right) \tag{2}$$

Your task is to classify each of the test sequences in the **Testing** data set according to the most likely speaker, $\hat{s}$:

$$\hat{s} = \underset{s=1,\ldots,S}{\text{argmax}} \ \log P\left(\tilde{X}|\theta_s\right) \tag{3}$$

Report the likelihoods of the five most likely speakers for each test utterance. Put these in files called `unkn_`N`.lik` for each test utterance N.

2.4 Experiments and discussion [10 marks]

Experiment with the settings of M and ϵ. For example, what happens to classification accuracy as the number of components decreases? What about when the number of possible speakers, S, decreases? You will be marked on the detail with which you empirically answer these questions and whether you can devise one or more additional valid experiments of this type.

Additionally, your report should include short hypothetical answers to the following questions:

- How might you improve the classification accuracy of the Gaussian mixtures, without adding more training data?
- When would your classifier decide that a given test utterance comes from none of the trained speaker models, and how would your classifier come to this decision?
- Can you think of some alternative methods for doing speaker identification that don't use Gaussian mixtures?

---

3 Speech Recognition

Speech recognition is the task of correctly identifying a word sequence given an input speech sequence X. Typically this process involves language models, dictionaries, and grammars. In this section we will

consider only a small subset of the acoustic modeling component. We will use the `Bayes Net Toolbox`, written for Matlab by Kevin Murphy, which is documented on-line at `http://bnt.googlecode.com`, along with some interface code that simplifies it for the case of continuous HMMs.

3.1 Continuous HMMs

Continuous HMMs are exactly like (discrete) HMMs except that the emission probability matrix is replaced by a vector of continuous probability density functions, one for each row of the former matrix. This is necessary because the set of outputs, K, is no longer discrete, but rather continuous. In this case, the outputs are all of the potential MFCC vectors. These vectors contain real numbers. For our specific purposes, we will assume that each of these probability densities is a Gaussian mixture model.

3.2 Phonetic annotation [10 marks]

We will require phonetic transcriptions for all test files. This will require you to use the WaveSurfer program[1]. You will also need the CMU dictionary again.

The `*.phn` files in */u/cs401/speechdata/Testing* have errors in them. Each file has one substitution error, one deletion error, and one insertion error. In this part of the assignment, you should find and correct these errors.

For each of the 10 files (i.e., `unkn_1.wav ... unkn_10.wav`), perform the following:

1. Open the wave file in WaveSurfer. When given the option, select "TIMIT transcription" as your configuration. The result should look similar to Figure 1, with the editable ".PHN" pane.
2. Open the associated text file and `*.phn` file.
3. Find each word from the text file in the CMU dictionary. Note the phonetic expansion of each word. You can consult Appendix C for a description of these phones.
4. Fix the transcription errors that you find, and ensure that all of the phone boundaries are accurate. You will be trained on marking phone boundaries in tutorial.
5. Save the resulting `*.phn` file — you will have to submit this.

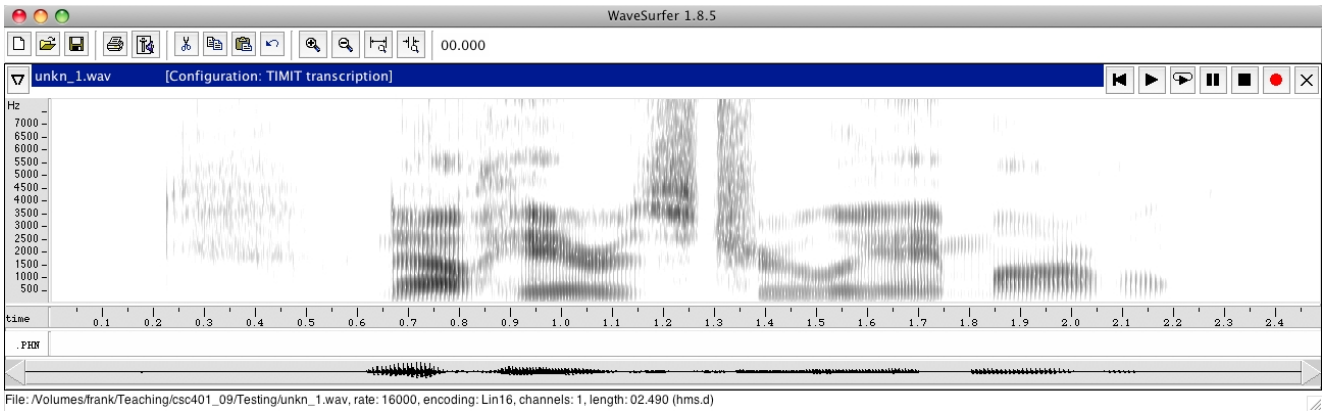

Figure 1: Example WaveSurfer window, including a spectrogram of `unkn_1.wav` .

[1]This can be obtained for home use at `http://www.speech.kth.se/wavesurfer/` .

Note: The transcription files produced by WaveSurfer will give the starting and ending *samples* of the associated waveform file. Since the waveform is sampled at 16kHz, and MFCC frames are windowed every 128 samples, you have to divide the numbers in these transcription files by 128 to match those in the `*.mfcc` files.

3.3 Training and decoding Hidden Markov models [15 marks]

Using the interface functions `initHMM` and `trainHMM` in */u/cs401/matlab*, write a simple program, `hmmTrain` , that can be used to initialize and train continuous hidden Markov models for each phoneme in the data set. You should not modify the interface functions, nor submit them along with your assignment. You won't need to study how they work either (this simply drags you into unnecessary details of the data structures of the BNT) — simply how to call them. Note that you will build a different model for each phoneme, and that you will train each of them with a collection of data from more than one speaker; hence these models will be *speaker-independent*.

Once you have trained the models, collect all the sequences from the test data given their respective `*.phn` files. Find the log likelihood of each phone sequence in the test data for each HMM phone model using the `loglikHMM` function. Report on the proportion of correct identifications of the phoneme sequences in the test data.

3.4 Log likelihoods revisited [10 marks]

Your worked on the last section using our log-likelihood function, which was written using BNT. Now you should write your own without BNT, assuming that you have a vector `Pi` of initial probabilities, a matrix `A` of transition probabilities, and parameters `W` , `Mu` and `Cov` for the Gaussian mixture models that govern the emissions from each state.

The stub file, */u/cs401/matlab/MYloglikHMM.m*, contains an interface between BNT and this simpler form for HMMs so that you can test your code with everything else. Add your code directly to this file and submit it.

3.5 Experiment and discussion [10 marks]

Again, experiment with changes to the parameters, e.g., $numDimensions$ and $numMixtures$, and the number of training examples. Your discussion will be marked by the same criteria as in Section 2.4.

3.6 Word-error rates [10 marks]

Imagine that we have now combined our phonetic HMMs into a larger speech recognition system that can recognize whole words and sentences. Develop a function `Levenshtein` in Matlab that computes the word-error rate between a test string and a reference string using Levenshtein distance. Assume that the cost of a substitution is 0 if the words are identical and 1 otherwise.

Assume that a speech recognition system produced the transcriptions for the *.wav files in the **Testing** directory and has written its hypotheses in the file **hypotheses.txt**, also in that directory. In that file, the i^{th} line is the hypothesis for `unkn_`i`.wav` .

Report on the word error rates between those transcriptions and the reference transcriptions (`*.txt`)

according to the following measures for each utterance, and for the entire data set:

$$SE = \text{prop. substituted words} = \frac{\text{\# Substituted words}}{\text{\# Reference words}}$$
$$IE = \text{prop. inserted words} = \frac{\text{\# Inserted words}}{\text{\# Reference words}}$$
$$DE = \text{prop. deleted words} = \frac{\text{\# Deleted words}}{\text{\# Reference words}}$$
$$\text{prop. total error} = SE + IE + DE$$

---

4 What to hand in

Your assignment should be submitted electronically only. You should submit:

1. All your code for `gmmTrain.m`, `hmmTrain.m`, `Levenshtein.m` and `MYloglikHMM` (including helper scripts, if any).
2. The files *unkn_*.lik*
3. Your correctly annotated output, `unkn_*.phn`.
4. Your discussion for both tasks, in a file called `discussion.txt`. Clearly demarcate the discussions of the two tasks — do not combine them into a single essay.
5. The `ID` file available from the course web-site.

Submissions missing any of the information requested in the ID file will not be marked.

Electronically: The electronic submission must be made from the CDF submission site. Do not `tar` or `compress` your files, and do not place your files in subdirectories.

5 Working at home

If you want to do some or all of this assignment on your home computer, you will have to do the extra work of downloading and installing the requisite software and data. You take on the risk that your home computer might not be adequate for the task. You are strongly advised to upload regular backups of your work to CDF, so that if your home machine fails or proves to be inadequate, you can immediately continue working on the assignment at CDF. When you have completed the assignment, you should try your programs out on CDF to make sure that they run correctly there. **A submission that does not work on CDF will get zero marks.**

A Appendix: File formats

Each utterance is represented by the five following file types:

`*.wav`	The original speech waveform sampled at 16kHz.
`*.mfcc`	The Mel-frequency cepstral coefficients obtained from an analysis of the waveform. Each line represents a 16ms frame of speech and consists of 14 space-separated floating point values.
`*.phn`	Frame-aligned phonetic transcription. Each line has the format [BEGIN] [END] [PHONE] where [BEGIN] and [END] are integers indexing the first and last frames of a span of the utterance with phonetic label [PHONE].
`*.wrd`	Frame-aligned word transcription. Each line has the format [BEGIN] [END] [WORD] where [BEGIN] and [END] are integers indexing the first and last frames of a span of the utterance with orthographic label [WORD].
`*.txt`	Orthographic transcription of the entire utterance. The line has the format [BEGIN] [END] [UTTERANCE] where [BEGIN] and [END] are the first and last frames of the utterance.

The MFCCs were obtained by applying 12 filters to windows of 256 consecutive samples of the speech waveforms, so each frame represents $256/16000 = 0.016$ seconds of speech. These windows are moved in increments of 128 samples, so frame f represents a window of speech beginning at $0.008f$ seconds. Each MFCC frame consists of 13 cepstral coefficients, including the 0^{th} order coefficient, as well as log energy and the first and second derivatives of those 14 variables.

All frame numbers in `*.phn, .*wrd` , and `*.txt` refer approximately to associated MFCC frame numbers.

B Appendix: Training Gaussian mixture models

Initialize θ

Initialize your model with randomly chosen parameters. Don't forget that they must be interpretable as probabilities.

Compute likelihood

Equation 4 is the observation probability for the m^{th} mixture component, given diagonal covariance matrices.

$$b_m\left(\vec{x}_t\right) = \frac{\exp\left[-\frac{1}{2}\sum_{n=1}^{d}\frac{\left(x_t[n]-\mu_m[n]\right)^2}{\sigma_m^2[n]}\right]}{\left(2\pi\right)^{d/2}\sqrt{\prod_{n=1}^{d}\sigma_m^2[n]}} \tag{4}$$

Given the feature vector $\vec{x}_t$ and parameters θ, the prior probability of the m^{th} Gaussian component is

$$p\left(m|\vec{x}_t,\theta\right) = \frac{\omega_m b_m\left(\vec{x}_t\right)}{\sum_{k=1}^{M}\omega_k b_k\left(\vec{x}_t\right)} \tag{5}$$

Update parameters

Given posterior probabilities computed using Equations 4 and 5, we want to update estimates for the mixture weights, means, and diagonal covariance matrices. These are accomplished thus:

$$\begin{aligned}
\hat{\omega}_m &= \frac{\sum_{t=1}^{T} p\left(m|\vec{x}_t,\theta\right)}{T} \\
\hat{\vec{\mu}}_m &= \frac{\sum_{t=1}^{T} p\left(m|\vec{x}_t,\theta\right)\vec{x}_t}{\sum_{t=1}^{T} p\left(m|\vec{x}_t,\theta\right)} \\
\hat{\vec{\sigma}}_m^2 &= \frac{\sum_{t=1}^{T} p\left(m|\vec{x}_t,\theta\right)\left(\vec{x}_t-\hat{\vec{\mu}}_m\right)^2}{\sum_{t=1}^{T} p\left(m|\vec{x}_t,\theta\right)}
\end{aligned} \tag{6}$$

In the third equation, the square of a vector on the right-hand side is defined as the component-wise square of each dimension in the vector.

C Appendix: TIMIT phone set

Tables 1, 2, and 3 list the permissible TIMIT phone symbols.

	symbol	example word	example transcription
stops			
	b	bee	BCL B iy
	d	dumb	DCL D ah m
	g	gum	GCL G ah m
	p	pea	PCL P iy
	t	tea	TCL T iy
	k	key	KCL K iy
	dx	muddy, dirty	m ah DX iy, dcl d er DX iy
	q	bat	bcl b ae Q
affricates			
	jh	joke	DCL JH ow kcl k
	ch	choke	TCL CH ow kcl k
fricatives			
	s	sea	S iy
	sh	she	SH iy
	z	zone	Z ow n
	zh	azure	ae ZH er
	f	fin	F ih n
	th	thin	TH ih n
	v	van	V ae n
	dh	then	DH e n
nasals			
	m	mom	M aa M
	n	noon	N uw N
	ng	sing	s ih NG
	em	bottom	b aa tcl t EM
	en	button	b ah q EN
	eng	washington	w aa sh ENG tcl t ax n
	nx	winner	w ih NX axr
semivowels and glides			
	l	lay	L ey
	r	ray	R ey
	w	way	W ey
	y	yacht	Y aa tcl t
	hh	hay	HH ey
	hv	ahead	ax HV eh dcl d
	el	bottle	bcl b aa tcl t EL

Table 1: TIMIT consonant phone-set.

	symbol	example word	example transcription
vowels			
	iy	beet	bcl b IY tcl t
	ih	bit	bcl b IH tcl t
	eh	bet	bcl b EH tcl t
	ey	bait	bcl b EY tcl t
	ae	bat	bcl b AE tcl t
	aa	bott	bcl b AA tcl t
	aw	bout	bcl b AW tcl t
	ay	bite	bcl b AY tcl t
	ah	but	bcl b AH tcl t
	ao	bought	bcl b AO tcl t
	oy	boy	bcl b OY
	ow	boat	bcl b OW tcl t
	uh	book	bcl b UH kcl k
	uw	boot	bcl b UW tcl t
	ux	toot	tcl t UX tcl t
	er	bird	bcl b ER dcl d
	ax	about	AX bcl b aw tcl t
	ix	debit	dcl d eh bcl b IX tcl t
	axr	butter	bcl b ah dx AXR
	ax-h	suspect	s AX-H s pcl p eh kcl k tcl t

Table 2: TIMIT vowel phone-set.

symbol	description
pau	pause
epi	epenthetic silence
h#	begin/end marker (non-speech events)
1	primary stress marker
2	secondary stress marker

Table 3: Other TIMIT codes.