Lecture-2

02/08/21

Roll no- 21410103 7

How to create a project in Visual Studio →

Step-1 → Click on "new project".
Step-2 → Choose win32 console application.
Step-3 → Give name to the project and its location.
Step-4 → Click "OK".
Step-5 → Click "finish".

or

Click "next" and choose whether you want an empty project or precompiled header. Then click "finish".

Pre-written code →

```
#include "stdafx.h"
int _tmain(int argc, _TCHAR* argv[]) {
    return 0;
}
```

Build or compile →

→ Go to "Build" option in toolbar and then choose "Build Solution".
→ Press F7.

Once build is successful, we shall run the program.

Run →

→ Go to Debug option in toolbar and click "Start without debugging".
→ Press F5.

The command prompt will appear and immediately go away, since there is nothing to be done.

First Prog.cpp →

```
#include "stdafx.h"
#include "stdio.h"
int _tmain(int argc, _TCHAR* argv[]) {
    int a,b,c;
    a=10;
    b=30;
    c=a+b;
    printf("%d+%d=%d", a, b, c);   return 0;
}
```

We shall follow the same process to build and run this program. But again the command prompt disappears immediately. So we shall do debugging or step by step execution.

Debugging the program →

We run the program line by line so that the command prompt shows the output after every line.

How to do it →

→ Go to Debug option and choose step over.

→ Press F10.

→ After pressing F10, multiple tabs will appear i.e. autos, call stack etc. These tabs help us to trace a particular instance of a program.

→ There will be a yellow arrow on the left side. It indicates the next line to be executed. In other words the program is suspended before reaching the line on which the arrow is present.

Pressing F10 twice →

```
#include "stdafx.h"
#include "stdio.h"
int _tmain(int argc, _TCHAR* argv[])
{
    int a,b,c;
⇨   a=10;
    b=30;
    c=a+b;
    printf("%d+%d=%d", a,b,c);
    return 0;
}
```

It means the program is executed till "int a,b,c". So we shall be able to see garbage values in a, b and c in Auto as follows.

Auto

Name	Value	Type
a	-858993460	int
b	-858993460	int
c	-858993460	int

(3)

→ If we again press F10,

⇨ a=10;

⇨ b=30;

The arrow will point to "b=30". It means a is now initialized to 10.

Autos will look as follows →

Auto		
Name	Value	Type
a	10	int
b	-858993460	int
c	-858993460	int

At the end of execution, Auto will look as follows →

Auto		
Name	Value	Type
a	10	int
b	30	int
c	40	int

→ We should initialize all variables to some initial value so that it doesn't contain garbage.

Using watch window →

We can put a break point at a location to suspend the program before it. We can do it in two ways.

→ Go to Debug and click on Toggle breakpoint.

→ Press F9.

Using this feature we can see each pass of bubble sort.

BubbleSort.cpp →

```
#include "stdafx.h"
#include "stdio.h"
int _tmain(int argc, _TCHAR* argv[])
{
    int a[] = {1,5,8,9,2,6,5,9,10,11};
    int i=0, j=0, temp=0;
```

④
```
for (i=0; i<=9; ++i)
{
    for (j=0; j<9-i; ++j)
    {
        if (a[j] > a[j+1])
        {
            temp = a[j];
            a[j] = a[j+1];
            a[j+1] = temp;
        }
    }
}
for (i=0; i<=9; ++i)
    printf("%d ", a[i]);
printf("\n");
return 0;
}
```

→ In the above program, we shouldn't write `for (int j=0; j<9-i; ++j)`. Since it is an inner loop, j gets initialized many times which we can avoid by declaring it at the top once.

→ The circle "O" indicates the break point.

Pressing F5 once →

Watch		
Name	Value	Type
a		
[0]	1	int
[1]	5	int
[2]	8	int
[3]	2	int
[4]	6	int
[5]	5	int
[6]	9	int
[7]	9	int
[8]	10	int
[9]	11	int

This table indicates the array state after one pass.
In the next pass
1 5 8 2 6 5 ... becomes
1 5 2 6 5 8 ... .

Final pass →

Watch		
Name	Value	Type
a		
[0]	1	int
[1]	2	int
[2]	5	int
[3]	5	int
[4]	6	int
[5]	8	int
[6]	9	int
[7]	9	int
[8]	10	int
[9]	11	int

Lecture – 3

03/08/21

Roll no- 21410103 7

Goal of research →
Goal is automatic recognition of speech by machine. The following disciplines have been applied to one or more speech recognition problems.

① Signal Processing
It is the form of analysis to characterize relevant information from speech signal. The information should be a true reflection of the actual phenomena. The signals should be robust and fault tolerant. The pre-processing and post-processing details are also taken into consideration.

② Physics (Acoustics)
Relationships between the physical speech signal and the physiological mechanism or the study of the same that produced the speech (vocal tract) and with which the speech was perceived (hearing tract).

③ Pattern Recognition
The set of algorithms to cluster data, to create one or more patterns or models on the basis of feature measurement of a pattern.

→ It involves generating patterns or finding out patterns and then storing them into a template or model and then use them later in decision making.

→ In speech processing, it is used to compare different speech patterns to determine their similarity.

④ Communication and Information Theory →
The procedures for estimating parameters of statistical models, the methods for detecting the presence of a particular speech pattern etc.

→ Information Theory is the scientific study of the quantification, storage and communication of digital information.
Here quantification refers to the speech signals and their parameters. Storage is done using models or templates.

⑤ Linguistics

The relationship between phonology, syntax, semantics and pragmatics.

Lecture – 4

06/08/21

Roll no- 21410103T

There are four different aspects in linguistics i.e.

1. Phonology

It is the study of sound patterns and their relationships that occur within languages. In other words, it studies how languages or dialects systematically organize their sounds.

2. Syntax

The legal arrangement or the legal way of writing the words in a language.

3. Semantics

The meaning of a word, phrase or text. The same words can give different meanings in different contexts.

4. Pragmatics

It deals with the sense or feel derived from the sentence or the context in which it is used.

⑥ Physiology

The understanding of the higher order mechanism of the human central nervous system that account for speech production and perception in human beings. We use Artificial Neural Networks to embed this mechanism information.

⑦ Computer Science

Study of efficient algorithms for implementing, in software or hardware, various methods of a speech recognition system.

⑧ Psychology

It deals with the understanding of the factors that enable humans to use speech recognition systems for practical tasks.

② Approaches for Speech recognition →

There are three approaches for speech recognition.

① Acoustic-Phonetic approach →

This approach directly exploits the individual sound properties in terms of broad acoustic features whose properties are relatively constant across words and speakers. This approach tries to find out what exactly happens, encodes it and puts into a system. So it works for limited resources also e.g. less number of speakers, sounds, words, limited RAM etc.

② Pattern Recognition approach →

It is a statistical or probabilistic approach. This approach tries to store the pattern in some model and then uses pattern recognition algorithm to map the input to the already stored data. e.g. Markov based modelling approach.

Good examples of this kind of system are- Siri, Alexa et

Interesting properties of Alexa:

It doesn't listen to everything. In better words, it doesn't process everything. It only processes the things told after a predefined keyword. Otherwise the device will burn out soon.

③ AI-based approach →

We try to mimic our human brain using huge numb of grid points or nodes and allow each of them to behave in a certain way. We try to manipulate the grid network by sending similar data many times hoping that the pattern will be captured and store So that it could be converged and extracted later.

It has neural based computing which handles matrice grid points, weightages, functions, back-propagation etc. to model non-linear speech. But the convergence is not guaranteed. Also a huge amount of training data is required.

(3)

Parts of Language →

① Vowels → a, e, i, o, u (in English)

② Consonants

Syllable → A part of a word which contains both vowels and consonants.

Fricative sounds → The kind of sounds that create noise are known has fricative sounds.

→ When we speak, air comes from lungs to vocal chords and we allow the vocal chords to vibrate. Then the sound comes through mouth.

→ We can change the sound by only changing the mouth configuration.

→ The speech recognition system heavily depends on how good the vowel recognition system is.

Various pronunciations →

'P' → starts by closing the mouth

'T' → Tongue or back of tongue is used ('D')

'K' → Back of throat ('G')

→ So the consonants allow us to stop the air and release again after configuration of mouth. So the complexity of speech increases.

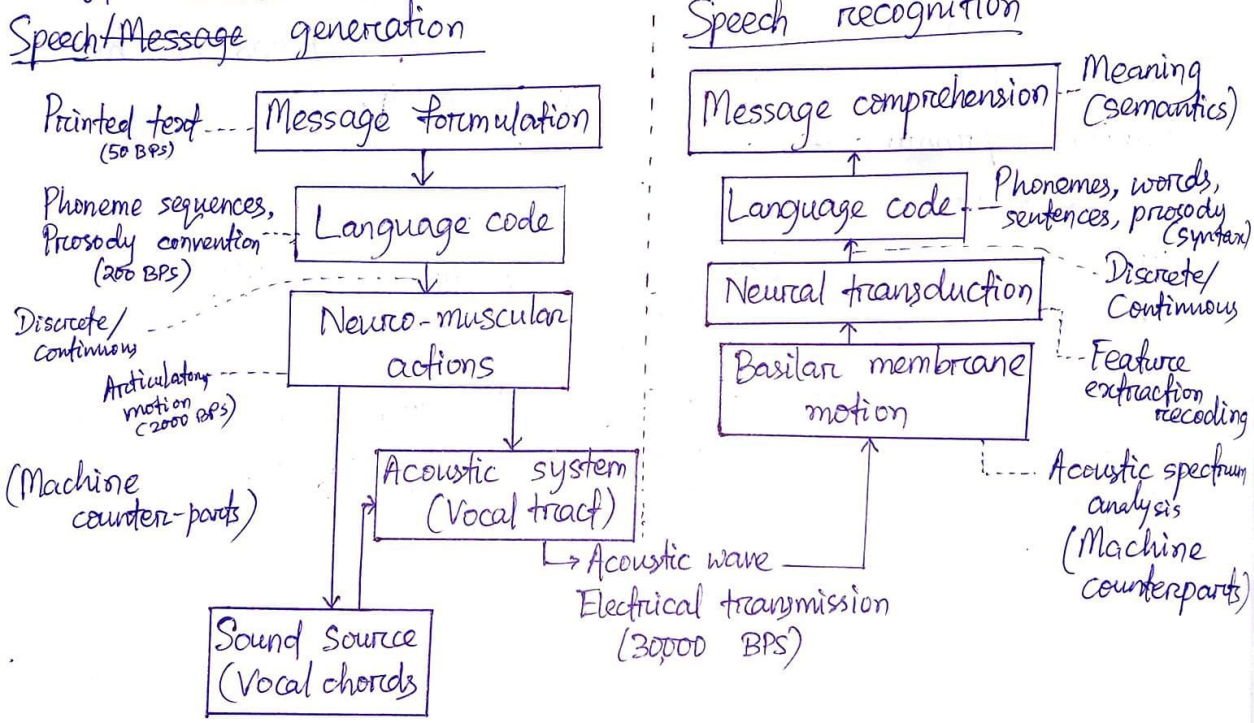
Lecture-5

09/08/21

Roll no- 214101037

Speech Production and Perception →

A Good understanding of the mechanics of producing and perceiving speech in human beings naturally leads to speech recognition by machine. Different classes of speech sounds and phonetics can be classified in terms of broad acoustic features whose properties are relatively invariant across words and speakers. This classification leads to straight-forward implementation of speech recognition algorithm based on sequential detection of sounds and sound classes.



Speech generation

Message formulation →

The production process begins when the talker produces or formulates a message. The message is meant to be transferred to the listener. The machine counter part to message formulation is creation of printed text expressing the words of a message.

Language code →

Then the message is converted to language code.

For example, we use Hindi or English to transmit a message. This includes converting the printed text to a set of phoneme sequences corresponding to the sounds that make up the words along with their durations, loudness and pitch accents.

Neuro-muscular actions →

After deciding the language code, a sequence of neuro-muscular actions should be performed in order to get acoustic form. The lungs act as the source of air for exciting the vocal mechanism. The muscle force pushes the air out of the lungs, through bronchi and trachea. When the vocal cords are tensed, the air flow causes them to vibrate producing the sound in the form of an acoustic signal as the final output. This must simultaneously control all aspects of articulatory motion including control of the lips, jaw, tongue and velum.

Speech recognition

Basilar membrane motion →

After the acoustic wave gets into the ear, the basilar membrane is set into motion. Then a group of nerves and fluid, convert this vibration into electro-chemical signals and transmit it to the brain.

Neural transduction →

The features from the signal are extracted. That means the signal is classified into various sounds like vowels, consonants etc.

Language code →

From the classification, the brain tries to detect which language the message belongs to. If the listener doesn't know the specific language, he won't be able to understand what is being spoken.

Message comprehension →

Finally the listener gets the exact message that is sent by the talker.

Speech sounds and features →

Based on the vocal cords, there is a three-state representation of speech: silence (where no voice is produced), unvoiced (where vocal cords do not vibrate and result in aperiodic or random waveform), voiced (where vocal cords are tensed and hence give quasi-periodic waveform.).

→ But it is very difficult to distinguish these three. We can't figure out where a speech is silent or weakly unvoiced, similarly, unvoiced or weakly voiced.

Phonemes →

Linguistically distinct speech sounds in a language is called phonemes.

Examples → /i/ in three, /a/ in Bob, /o/ in ox, /u/ in put etc.

Lecture-6

10/08/21

Roll no- 21410103 7

To check the ambient room noise →
Press F10.
→ A red progress bar will indicate the average noise level in dB (decibel).
→ Any number <-30 is good. Numbers > -30 indicates high noise.
Press F10 again to stop the process.
→ Too much noise correction is not good. It might miss the beginning part of the speech.

To record a speech →
① Sampling rate → set it to 16000 samples per second. That means we'll get 16000 values per second.
② Channel → set it to mono.
③ Resolution → set it to 16-bit. That means all the values will be 16-bit numbers.
→ The background noise will overlay the speech.

Analysis of the waveform →
→ After magnifying the waveform, we can see the waves are periodic. This kind of wave indicates that a vowel is spoken. But we can't find any such pattern in case of fricatives.
→ Fricative sounds have less energy compared to vowels.

To save the recording →
→ Click on File.
→ Click on save as.
→ Change the type to ASCII Text Data(*.txt).
→ Give some name and save it.

Features of speech →
① Energy →

$$E = \frac{1}{N}\sum_{i=1}^{N} x_i^2$$

Energy is the sum of squares of the values that

we got from the waveform. We shall not consider all the values at once. We can choose a small value like $N = 100$ or $N = 200$.

→ This feature tells us how loudly we have spoken.
→ If nothing has been spoken, energy is very low.
If vowel has been spoken, energy is very high.
But fricative can have both high or low energy.

② Zero Crossing Rate (ZCR) →

It is the number of times the waveform cuts the X-axis.

logic: if $x_i \geq 0$ & $x_{i+1} < 0$ ++zcr
if $x_i < 0$ & $x_{i+1} \geq 0$ ++zcr

→ For fricatives we get very high zcr. Vowels give low zcr.

To distinguish YES and NO →
→ YES has a trailing fricative. We can check if there is any fricative at the end.

Assignment - 1 →

Deadline → ~~13th~~ 16th August

Write a program (preferably in C) to detect two words i.e. YES and NO.

Lecture-7

13/08/21

Roll no- 214101037

Things we need to check before recording →

① Recording hardware →

We need to check the microphone properly. Sensitive microphones record the background noise very loudly. We need to choose a microphone which is not very sensitive and speech should be recorded clearly.

② DC shift analysis →

If the machine/laptop is on charge, and we are grounded and we touch the body of the machine. we can feel a little bit vibration. This happens when earthing is not properly done. This is usually dangerous.

For speech processing, on cool edit software, if we see the signal when there is no microphone attached, this is called DC shift because of the current leak to the audio software of the system. So we need to ensure that DC shift is either eliminated or minimized. So we need to subtract the shift value from each subsequent samples we take in future for that particular system. This process is called DC shift correction. We do it so that the signal should be at 0 when microphone is off.

③ Normalization (wave normalization) →

We need to decide the maximum depth of the waveform on both the directions. We can fix it at 5000 (-5000 for the other side). So if we speak too loud, we need to reduce the values and if we speak soft, we need to amplify the values. Otherwise while calculating the energy, since we square the numbers and add them, we might get overflow

or underflow problems. So we can multiply the values by (5000 / maximum) or (-5000 / minimum) for positive and negative values respectively. It will put the waveform within the range of 5000 to -5000 on both sides respectively.

Finding Ambient noise / Background noise →

We can record without speaking anything and get the energy and zcr values. There might be a few systems, when we hit the record button, in the initial few parts there will be a random noise which might give very high energy, close to vowels.

→ To avoid this, either we can get a very good quality microphone, or we can throw away the first few frames and start considering from 4th or 5th frame.

→ After throwing away initial few frames we can start ambient noise calculation.

→ When we get a clear deviation in energy, that will conclude that something is being spoken starting from that point.

→ We can throw away small words which have a very few frames.

→ We can't cut away the noise strongly since we might end up losing the word itself.

Assignment procedure →

① Check DC shift. If any non-zero value is obtained, it should be subtracted from all the data points.

DC shift calculation logic →

```
int sum = 0, x = 0;   float dcShift = 0.0;   int count=0;
read numbers from file into x;
~~sum+=x;~~   sum = sum + x;
keep counting the no of x values in count;
dc shift = sum / count;
```

(3)

(2) Normalize the data.
We need to find out the normalization factor and multiply it to all the data points.
Logic →
int maxData =0, minData=0; data =0;
read numbers into data;
Keep updating maxData and minData;
return 5000 /((maxData+minData)/2);

(3) Calculate the energy and zcr for sample size of 100.

(4) 20-25 samples from the beginning can give an idea of the ambient noise.

(5) We need to check at which point energy is deviated by more than 110%-120% of the ambient noise. That will be the starting point.

(6) Similarly, after starting point, when it goes below the ambient noise, that will be the ending point.

(7) To get a better marker, we can use idea of buffer. That means if 5-10 continuous samples satisfy the above two properties then only decide the starting and ending points.

(8) ZCR threshold = 30-40. (assumption)
Average zcr of last 40% of NO is 1-10, but that of YES is >50. Hence the threshold can be anything greater than 15.

(9) DC shift ≤0.0005 ≃ 0.

Lecture - 8

16/08/21

Roll no - 214101037

Phonemes →

These are the basic units of sound in any language that is spoken. Phonemes can be represented by phonetic alphabets. A given language is broken into smaller units for studying its characteristics by linguistics. The smallest unit of speech is termed as phonemes. e.g. for American English there are 48 phonemes. For every language, there is a fixed set of phonemes.

Classification →

① Vowels ② Dipthongs ③ Semi-vowels ④ Consonants

① Vowels →

Vowels have very steady sound and the vocal chords are continuously vibrating. Different vowels have different waveforms. By looking at them we can distinguish the vowels. The waveforms are a bit different people due to the vocal chords construction. Most recognition systems rely heavily on vowels. Vowels are long in duration and spectrally very well defined.

Vowels are further classified as -

FRONT → i (IY)
I (IH)
ɛ (EH)
æ (AE)

MID → ɑ (AA)
ɜ (ER)
ʌ (AH)
ə (AX)
ɔ (AO)

② BACK→

u (UW)
U (UH)
O (OW)

Examples →

① Th_y n_td s_gn_f_c_nt _mpr_v_m_nts _n th_ c_mp_ny's _m_g_, s_p_rv_s__n, th__r w_rk_ng c_nd_t__ns, b_n_f_ts _nd _pp_rt_n_t__s f_r gr_wth.

Here vowels are omitted. But they can be guessed with a small amount of efforts. The sentence says "They noted significant improvements in the company's image, supervision, their working conditions, benefits and opportunities for growth."

② But if we remove consonants, it is almost impossible to guess them.

A__i_u_e _o_a___ _a_ __a_e_ e__e__ia__
__e _a_e, _i__ __e __o_e_ o_ o__u_a_io_a_
e___o___ __i_____ _e__ea_i__.

Which says
"Attitude towards pay stayed essentially the same, with the scores of occupational employees slightly decreasing."

→ While speaking, vowels are produced by exciting an essentially fixed vocal tract shape with quasi-periodic pulses of air caused by the vibration of the vocal chords.

→ There are several ways to characterize vowels, including the typical articulatory configuration required to produced the sounds, typical wave-form plots and typical spectrogram plots.

→ A typical way to distinguish is the tongue hump position i.e. front, mid or back.

→ The front vowels show a pronounced high frequency resonance. The mid vowels show a balance of energy over a broad frequency range. The back vowels show a predominance of low-frequency spectral information.

How to analyse vowels in different forms →

→ Record a vowel in cool edit.
→ If we amplify the waveform we can see that a vowel has been spoken.
→ The X-axis shows time and Y-axis shows the amplitude.
→ This is also called a time plot.

To change the plot →

→ Click on "view".
→ Click on "spectral view".
→ We can change the view from colours to black and white and vice versa by clicking on "options" → "settings" → "colours" → choose a scheme.
→ The view is called a spectrogram. This is a 3D plot. X-axis and Y-axis show time and frequency respectively. Colours are the third dimension.
→ The colours signify the energy.
→ If we select any part, the colors are reversed.

Frequency analysis →

→ click on "Analyze".
→ click on "Frequency analysis".
→ A frequency window will appear.
→ Choose 256 FFT (Fast Fourier Transform).
→ If we play the audio, the waveforms start moving.
→ The waveforms have got some fixed peaks

(4) that do not move much. That is called "formant frequency.

→ For a given waveform, if the sampling rate is n samples per second, then the maximum frequency resolution is $\frac{n}{2}$. e.g. → 16000 sampling rate gives 8000 frequency resolution.

→ Different vowels give different spectral view.

→ The frequency ware represents a vertical line of the spectral view at a specific time point.

→ The front vowels show ~~well~~ a relatively high second and third formant frequency. The mid vowels show well-separated and balanced locations of the formants and the back vowels show almost no energy beyond the low frequency region with low first and second formant frequencies.

(2) **Dipthongs →**

Dipthong is a sliding sound which sounds like vowels. Dipthong starts from the configuration of one vowel and transitions towards another vowel. This is monosyllabic or single unit speech sound but has a sliding property. In other words, it starts at or near the articulatory position of one vowel and moves towards the position of another.

Examples → /a^y/ → (AY)
/$ɔ^y$/ → (OY)
/a^w/ → (AW)
/e^y/ → (EY)

The challenge is to characterize dipthongs. It is tough to do this, even by the recognition system

Buy → /a^y/ → transition from a to y.

Bait → /e^y/ → transition from e to y.

(5)

/j^{u}/ → you → transition from j to u
/a^{w}/ → down → transition from a to w
/o^{a}/ → boat → transition from o to a
/ɔ^{y}/ → boy → transition from o to y

Dipthongs are produced by varying the vocal tract smoothly between vowel configurations appropriate for the dipthong.

③ Semi vowels →

These sounds are almost like vowels. We move our tongue by flattening it and allowing the air to pass through the sides of the tongue. These are also called laterals. These are very difficult to characterize. Some of the sounds are not readily decoded in terms of features.

Liquids → w (W)
l (L)

glides → r (R)
y (Y)

④ Nasal consonants →

These sounds are created when we block vocal way and a good amount of air passes through the nose. Head also vibrates a bit while producing these sounds. We can't breathe when producing these sounds. These are also pretty tough to characterize.

e.g. → /m/, /n/ & /ŋ/

→ The spectral view of the dipthongs can be seen with a transition from one energy pattern to other. Also, in the waveform view, two different kind of periodical waveforms can be seen.

Lecture - 9

17/08/21

Roll no - 21410103 7

Nasal sounds come through nasal cavity.
Chanting of "AUM" → All the vowels are uttered and our brain starts vibrating. It is good for health, if we do it properly. This is beneficial for

① Stomach health - studies suggest that chanting om on a regular basis relaxes the muscles of stomach.

② Calms mind - anger can also be controlled.

③ Reduce stress and anxiety - helps to focus or concentrate.

④ Other benefits - The vibration starts from vocal chords and ends up in sinuses. So it can also give relief from sinus problems.

⑤ Unvoiced fricatives →

These are produced when a steady flow of air in the vocal tract becomes turbulent in the region of a constriction. These sounds have less energy but these are highly noisy. The location of the constriction determines the type of sound. Unvoiced fricatives indicate that the vocal cords are not vibrating.

Examples → /f/, /θ/, /s/, /sh/

constriction location

/f/ → near the lips
/θ/ → near the teeth
/s/ → it is near the middle of the oral tract
/sh/ → near the back of the oral tract

The waveforms and spectrograms of fricatives give no pattern at all and have a high zcr value.

⑥ Voiced fricatives →

While producing these sounds, initially the air flow is blocked. Voiced fricatives indicate that vocal

cords are in action but it is still noisy. So it will be a bit periodic with no pattern since constriction is still there.

/v/, /th/, /jh/, /ʒ/, /zh/.

/v/ → at lips
/th/ → near teeth
/z/ → middle of oral tract
/zh/ → back of oral tract

} constriction location

Voiced and unvoiced stops →

These are generated by not opening the mouth at all. The air flow is completely blocked. Then lungs force air to oral tract and the air is released.

Voiced stops → The constriction is somewhere in the oral tract. Pressure is built and then suddenly released. Vocal cords are also vibrated.

/b/, /d/, /g/.

/b/ → lips
/d/ → back of teeth
/g/ → back of tongue

} constriction location.

Unvoiced stops → In this case the major exception is the vibration of vocal cords (they do not vibrate for unvoiced sounds.

/p/, /t/, /k/ are the respective counter parts of voiced stops and the constriction locations are also similar.

→ In Hindi, the vowels are segregated. (similar in Odia).
→ For consonants, if we go row wise there are diffent properties and for columnwise there are different properties.
→ The environment affects the pronunciation a lot.

(3)

→ We can see a small waveform before opening the mouth. That is the vocal cord vibrations got captured. The microphone picked the vibration from the neck. This is called "voice bar".

→ The waveforms are periodic for vowels.

→ Female voices have higher pitch and the time of periodicity is less.

→ All the vowels have different waveforms. The periodicity and waveforms can be used to detect the vowels.

Lecture -10 20/08/21

Roll no - 21410103 7

Frequency Domain Approach →

We need to cut the frequencies into chunks and analyse which chunks have high energy. It tries to represent a signal by sums of sinusoidal or complex exponentiations. E.g → Tuning fork experiment. It generates a tone which has got a specific property. 8000 Hz means the tuning fork will vibrate 8000 times per second. Tuning forks vibrate with a single frequency but when we speak, there are multiple frequencies. E.g → for low pitched sounds the frequency is less and vice versa.

This centers around one formula of fourier analysis. Fourier analysis or fourier transform converts a time domain signal to a frequency domain signal

Fourier Transform →

Let x(n) be a discrete time signal.

Then fourier transform of x(n) is given by -

$$x(e^{iw}) = \sum_{n=-\infty}^{\infty} x(n) \cdot e^{-jwn} \quad \text{---------- ①}$$

$$x(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{jw})\, e^{+jwn} \quad \text{---------- ②}$$

where w is the frequency

In eq-①

The same speech signal (x(n)) is being multiplied to a factor. The result is a complex number. If we take its mod, we shall get the fourier coefficient i.e. the magnitude of the frequency.

In eq-②

Integrate the fourier coefficient to get back the original signal.

Resonance → Every material in earth has a natural frequency. If we hit an object with a frequency and we keep increasing it, at some point the object will vibrate heavily. That point is the natural frequency of the object.

So, if we speak at the natural frequency of the vocal cords, huge amount of energy is generated.

Formant frequency → It characterizes the frequency components well. This is the frequency which are set up along with the fundamental frequency.

$\sum_{n=-\infty}^{\infty} x(n)$ is an unbiased sum without any frequency component.

The extra term which is added, will give sinusoidal form. It is dependent on w. So this function is cyclic in nature.

A sufficient condition for the existence of the fourier transform is

$$\sum_{n=-\infty}^{\infty} |x(n)| < \infty$$

Discrete Fourier Transform →

For a discrete signal $x(n)$, if the signal is periodic with period N i.e. $\tilde{x}(n) = \tilde{x}(N+n)$; $-\infty < n < \infty$.

Then $\tilde{x}(n)$ can be represented by a discrete sum of sinusides

$$\tilde{X}(K) = \sum_{n=0}^{N-1} \tilde{x}(n) \cdot e^{-j \cdot \frac{2\pi}{N} \cdot kn} \quad \text{——— (3)}$$

$$\tilde{x}(n) = \frac{1}{N} \sum_{K=0}^{N-1} \tilde{X}(K) \cdot e^{+j \cdot \frac{2\pi}{N} \cdot kn} \quad \text{——— (4)}$$

$$w = \frac{2\pi}{N} \cdot K$$

Lecture - 11

23/08/21

Roll no - 21410103 7

Z- Transform →

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)\, z^{-n} \quad \text{——— (5)}$$

$$x(n) = \frac{1}{2\pi j} \int_C X(z) \cdot z^{n-1}\, dz \quad \text{——— (6)}$$

If we replace z with a proper value, we can get a different transform.
If we take $z = e^{j \cdot \frac{2\pi}{N} \cdot k}$, we get fourier transform.
Z- transform is a general purpose transform.
f_0 is the fundamental frequency, or pitch.
This is the lowest frequency of highest magnitude of vowels.
When we talk about musical instruments like harmonium or keyboard, there are 7/12 tones but many keys. So they represent the same tone with diffent pitches.

Time Domain Speech Analysis →

Linear predictive coding or LPC →

It is simple and mathematically proven.
Why LPC? It's ancient!
→ It is a good model specifically for periodic parts of speech.
→ It leads to good source vocal tract separation. It means, the source component (lungs and vocal chords), vocal tracts (tongue, teeth, lips etc) are separated well by this coding.
→ It is a mathematical model. It becomes convenient to manipulate the formulae, proofs and theory.
→ It is a proven method.

Let $S(n)$ be the speech signal at time 'n'. This can be approximated as a linear combination of the past p samples such that

$$S(n) \simeq a_1 S(n-1) + a_2 S(n-2) + \cdots + a_p S(n-p) \quad \text{——— (1)}$$

where a_i ; $i = 1, 2, \cdots, p$ are constants.

Convert (1) to an equality by including a term called (excitation term) $G \cdot u(n)$ giving

$$S(n) = \sum_{i=1}^{p} a_i S(n-i) + G \cdot u(n) \quad \text{——— (2)}$$

We can get the next point by applying this formula on the past p points or samples, with the help of some constants or coefficients. But this is an approximation.

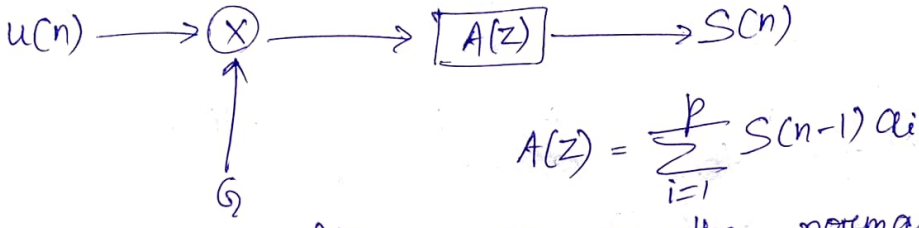
During unvoiced and transient regions of speech, LPC model is less effective than for voiced regions, but it still provides an acceptably useful model for speech recognition purposes.

In eq-(2), $u(n)$ is the normalized excitation and G is the gain term.

Lecture - 12

24/08/21

Roll no - 214101037



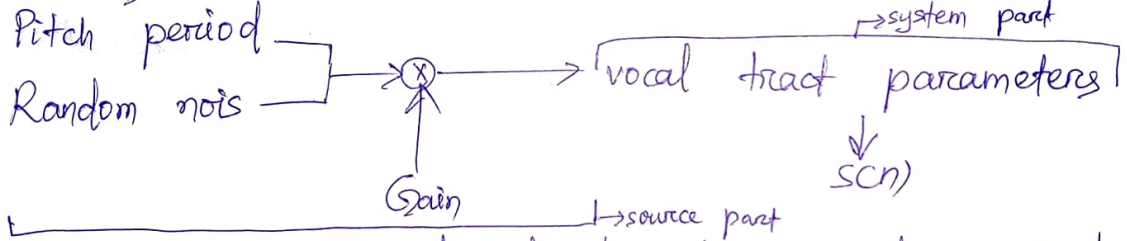
$$A(Z) = \sum_{i=1}^{p} S(n-1)\, a_i$$

A(Z) is a filter. u(n) is the normalized excitation and G is the gain term. We put them together and pass through A(Z) which is Linear Predictive Coding filter and we get the prediction s(n).

→ Our physiology has helped to design this flow.

→ The flow of air and the amount of pressure or energy put are governed by the gain term.

→ u(n) tells us whether the vocal cords are vibrating or not. So, this is called excitation term.



We can use vocal chords to generate voiced signals, or we might not use it, which gives random noise. Then we apply gain to it. Then the vocal tract will articulate the speech which will be generated. So the last p signals are good parameters.

Let's consider

$$\hat{S}(n) = \sum_{i=1}^{p} a_i\, S(n-i) \qquad \text{---(3)}$$

We dropped G.u(n). This equation gives us the ~~source~~ system part of the speech. It means we are trying to find the details from only the system part and we dropped the source part.

In other words we are generating speech from vocal tract parameters only.

Now, the prediction error is given by

$$e(n) = S(n) - \tilde{S}(n)$$

$$= S(n) - \sum_{i=1}^{p} a_i S(n-i)$$

Here, a_i; $i = 1, 2, \ldots, p$ for a short segment of speech signal.

So we are not taking the entire speech sample and only a part of it. The source part is now included in the error.

We have to find the values of the coefficients a_i.

Let us take

$$S_n(m) = S(n+m)$$

$$e_n(m) = e(n+m)$$

We shall vary one variable.

Now we seek to minimize the MSE (Mean Squared Error) signal at time n i.e.

$$E_n = \sum_m e_n^2(m)$$

here E_n is the mean squared error

$$= \sum_m \left[S_n(m) - \sum_{K=1}^{p} a_K S_n(m-K) \right]^2$$

Differentiate partially with respect to each a_i and set the equation to zero.

$$\frac{\partial}{\partial a_i}(E_n) = 0$$

$$\Rightarrow \frac{\partial}{\partial a_i} \sum_m \left[S_n(m) - \sum_{K=1}^{p} a_K S_n(m-K) \right]^2 = 0 \; ; \; i = 1, 2, \ldots, p$$

$$\Rightarrow \frac{\partial}{\partial a_i} \sum_m \left[S_n^2(m) + \left\{ \sum_{K=1}^{p} a_K S_n(m-K) \right\}^2 - 2 \cdot S_n(m) \cdot \sum_{K=1}^{p} a_K \cdot S_n(m-K) \right] = 0$$

(we did $(a-b)^2 = a^2 + b^2 - 2ab$)

Since $S_n(m)$ doesn't have any a_i term in it, the differentiation of $S_n^2(m)$ will be 0.

(3)

$$\Rightarrow \frac{\partial}{\partial a_i} \sum_m \left[\sum_{K=1}^{p} a_K^2 S_n^2(m-K) + 2 \sum_{\substack{K=1 \\ i \neq K}}^{p} a_i a_K \cdot S_n(m-i) S_n(m-K) \right]$$

$$= \sum_m 2 \cdot S_n(m) \cdot \frac{\partial}{\partial a_i} \left\{ \sum_{K=1}^{p} a_K S_n(m-K) \right\}$$

We expanded the b^2 term and took $-2ab$ to the right hand side.

$$\Rightarrow \sum_m 2 a_i S_n^2(m-i) + 2 \sum_m \sum_{\substack{K=1 \\ i \neq K}}^{p} a_K S_n(m-i) S_n(m-K)$$

$$= 2 \sum_m S_n(m) \cdot S_n(m-i)$$

For the first term on the left hand side, only a_i term will remain and rest $p-1$ terms become zero. After differentiation, a_i^2 becomes $2a_i$.

For the second term on the left hand side, a_i term is differentiated which gives 1.

For the right hand side, similar to the first term in left hand side, only term having a_i will remain and others will become zero. Then a_i becomes 1 after differentiation.

$$\Rightarrow \sum_m a_i \cdot S_n^2(m-i) + \sum_m \sum_{\substack{K=1 \\ i \neq K}}^{p} a_K S_n(m-i) S_n(m-K)$$

$$= \sum_m S_n(m) \cdot S_n(m-i)$$

2 is cancelled off on each side.

$$\Rightarrow \sum_{K=1}^{p} a_K \sum_m S_n(m-i) S_n(m-K) = \sum_m S_n(m-0) \cdot S_n(m-i)$$

Since we have written $i \neq K$ in the condition of the second term on the left hand side and the first term contains everything with $K=i$, so we can merge both the terms on

(4)

the left hand side. Also on the right hand side, m can be written as m-0.

$$\Rightarrow \sum_{k=1}^{p} a_k \phi_n(i,k) = \phi_n(0,i)$$

where $\phi_n(i,k) = \sum_m S_n(m-i) \cdot S_n(m-k)$

We call them the set of normal equations.

Lecture - 13

27/08/21

Roll no - 21410103 7

There are two techniques to solve the set of normal equations. It helps us to predict the values of a_i and the nth sample, given the past p samples.

When we solve the equations, the determinant value should not be 0.

① Auto correlation method →

In all kind of methods, we have constraints to ensure that the solution works for real world too.

Let us assume that the waveform segment $S_n(m)$ is identically 0 outside the interval 0 and N-1 i.e. $0 \leq m \leq N-1$.

The value of m (which is a short interval of speech) should be 20 ms i.e. 320 sample size for 16000 samples per second.

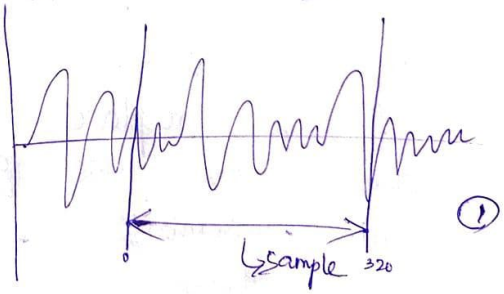
The values beyond the region of 320 samples, should go to 0. We need to multiply a factor to maintain this.

$$S_n(m) = w(m) \cdot s(n+m) \quad \text{——— (5)}$$

where $w(m)$ is a finite length window.

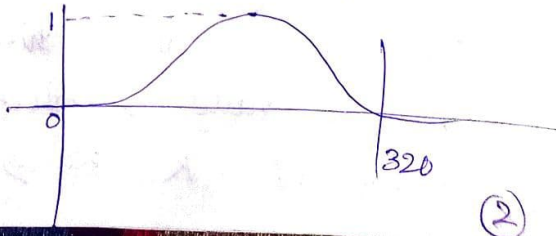
We clamp a window on the top of the signal. This window is usually a hamming window.

We need to multiply the hamming window to the entire sample i.e. all the points in the region, as shown in the picture ①.



①

Picture - ② shows how the values are changed after multiplication. This is a cosine function.



②

Now we shall check the benefits of auto-correlation method. So,

$$\phi_n(i,k) = \sum_{m=0}^{N+P-1} S_n(m-i)\, S_n(m-k)$$

Any value of P, will result in 0 value, since we have already made everything 0, outside the range (0, N-1).

let $m-i = y \Rightarrow m = y+i$

$$\phi_n(i,k) = \sum_{y=0-i}^{N+P-1}$$

But anything beyond 0 gives 0 so we can write $y=0$. P can also be ignored for same reason.

i.e. $$\sum_{y=0}^{N-1-(i-k)} S_n(y).\, S_n(y+i-k) \qquad \text{——— (6)}$$

We subtracted i-k from the upper limit, since that is an extra term in $S_n(y+i-k)$ for which the upper limit will go beyond N-1.

What is correlation?

It is the degree of agreement between two series X and Y, or the degree of synchronization. Both the series are given by some numbers.

If both the series are going together, there is some correlation. Otherwise the agreement is meaningless. The series should either be increasing or decreasing.

If we replace Y with X i.e. have a single series, we are going to correlate X with X. This resembles to energy calculation since we were taking squares of the amplitude values.

If we are shifting the second X series by K values, that is an auto correlation by a lag of K values.

After shifting, we calculate the normal correlation.

$$R_k(X) = \frac{1}{N} \sum_{i=1}^{N} x_i \, x_{i+k}$$

This is called auto correlation of lag k.
If we take $k=0$ i.e. no lag, that becomes the average energy.
If we keep shifting the series continuously, the energy value will keep in reducing. But when the pitch period is touched i.e. $k = 50$ or 60, there will again be a spike in energy for vowels, since vowels give periodic waveforms. There both the series become equal.

Eq-⑥ is an short term auto correlation function with lag $i-k$, where $\phi_n(i,k) = R_n(|i-k|)$

$$R_n(k) = \sum_{m=0}^{N-1-k} S_n(m) \cdot S_n(m+k)$$

Also $R_n(k)$ is an even function i.e.

$$\phi_n(i,k) = R_n(|i-k|)$$

Now eq-④ becomes $\sum_{k=1}^{p} a_k R_n(|i-k|) = R_n(i)$

where $i = 1, 2, 3, \cdots, p$

In matrix notation, we get

$$\begin{bmatrix} R_n(0) & R_n(1) & \cdots & R_n(p-1) \\ R_n(1) & R_n(0) & R_n(1) \cdots & R_n(p-2) \\ R_n(2) & R_n(1) & R_n(0) \cdots & R_n(p-3) \\ \vdots & & & \\ R_n(p-1) & R_n(p-2) & \cdots & R_n(0) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_p \end{bmatrix} = \begin{bmatrix} R_n(1) \\ R_n(2) \\ \vdots \\ R_n(p) \end{bmatrix}$$

$p \times p$ $\quad p \times 1$ $\quad p \times 1$

$R_n(-1)$, $R_n(-2)$ and similar values are written after taking their absolute values. Those terms are underlined.

Lecture - 14

30/08/21

Roll no - 214101037

→ The rows of the matrix should be linearly independent and the rank should be equal to p.

→ This matrix is symmetric and it is called toeplitz matrix.

→ If we go from top to bottom diagonally, all the diagonals have same elements within themselves.

→ Toeplitz matrix means the matrix is symmetric in nature and the diagonals are same. These properties should be exploited to get the most efficient solution.

→ We need to compute the determinant of this matrix.
Under what conditions, the determinant is zero?
The energy should be zero to make the determinant zero. If energy is 0 then the signal is not present. So if the energy is 0 then skip the frame.

→ From engineering perspective, we need to set a threshold for $R_n(0)$ so that any frame having energy less than the threshold can be skipped.

Levinson - Durbin algorithm →

This is recursive.

$$E^{(0)} = R(0)$$

$$K_i = \left[R_i - \sum_{j=1}^{i-1} \alpha_j^{(i-1)} \cdot R(i-j)\right] / E^{(i-1)} \quad ; \quad 1 \le i \le p$$

$$\alpha_i^{(i)} = K_i$$

$$\alpha_j^{(i)} = \alpha_j^{(i-1)} - K_i \cdot \alpha_{i-j}^{(i-1)} \quad ; \quad 1 \le j \le i-1$$

$$E^{(i)} = (1 - K_i^2) \cdot E^{(i-1)}$$

② $R(0), R(1), \cdots, R(p)$ are the input.
$R(0)$ is the energy and $R(p)$ is the auto-correlation coefficient with lag p.
For $K_i = \frac{R_i(j)}{R(0)}$ since j going from i to 0 has no meaning.
These are solved for $i = 1, 2, 3, \cdots, p$.
In the computation of $i=1$, the summation is dropped.
$\alpha_i = \alpha_i^{(p)}$ for $i = 1, 2, 3, \cdots, p$ is the final answer. That means pth iteration gives the final answer.

→ $N = 320$, $p = 12$ (rule is to use order four more than the sampling rate, but we shall stick to 12).
$N = 320$ means 20 ms of speech.

Assignment →

Consider a predictor of order $p=2$ and assume the auto correlation vector $R = \{r_0, r_1, r_2\}$. Use Durbin's algorithm to solve for the coefficients a_i's $(i=1,2)$ in terms of r_i's. Check your answer by solving the matrix equation

$$\begin{pmatrix} r_0 & r_1 \\ r_1 & r_0 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix}$$

The time complexity is $O(p^2)$. But matrix solution goes $O(p^3)$. Hence Durbin's algorithm is faster by an order of magnitude.

Answer →

Durbin's algorithm →

1. $E^0 = r_0$
2. for $i=1$
2. $K_1 = \frac{r_1}{r_0}$

3. $\alpha_1^{(1)} = K_1 = \frac{r_1}{r_0}$

4. $E^{(1)} = (1-K_1^2) \cdot E^0 = \left(1 - \frac{r_1^2}{r_0^2}\right) r_0 = \frac{r_0^2 - r_1^2}{r_0}$

For $i=2$

5. $K_2 = \left[r_2 - \sum_{j=1}^{1} \alpha_j^{(i-1)} \cdot R(i-j)\right] \Big/ E^{(2-1)}$

$= \left[r_2 - \alpha_1^{(1)} r_1\right] / E'$

$= \left(r_2 - \frac{r_1}{r_0} \cdot r_1\right) \Big/ \frac{r_0^2 - r_1^2}{r_0}$

$= \dfrac{\dfrac{r_0 r_2 - r_1^2}{r_0}}{\dfrac{r_0^2 - r_1^2}{r_0}}$

$= \frac{r_0 r_2 - r_1^2}{r_0^2 - r_1^2}$

6. $\alpha_2^{(2)} = K_2 = \frac{r_0 r_2 - r_1^2}{r_0^2 - r_1^2}$

7. $\alpha_1^{(2)} = \alpha_1^{(1)} - K_2 \cdot \alpha_1^{(1)}$

$= \frac{r_1}{r_0} - \left(\frac{r_0 r_2 - r_1^2}{r_0^2 - r_1^2} \cdot \frac{r_1}{r_0}\right)$

$= \frac{r_1}{r_0} - \frac{r_0 r_1 r_2 - r_1^3}{r_0 (r_0^2 - r_1^2)}$

8. $E^{(2)} = (1-K_2^2) \cdot E^{(1)}$

$= \left[1 - \frac{(r_0 r_2 - r_1^2)^2}{(r_0^2 - r_1^2)^2}\right] \cdot \left(\frac{r_0^2 - r_1^2}{r_0}\right)$

(4) Verification →

$$\text{LHS} = \begin{bmatrix} r_0 & r_1 \\ r_1 & r_0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$

$$= \begin{bmatrix} r_0 & r_1 \\ r_1 & r_0 \end{bmatrix} \cdot \begin{bmatrix} \frac{r_1}{r_0} - \frac{r_0 r_1 r_2 - r_1^3}{r_0(r_0^2 - r_1^2)} \\ \frac{r_0 r_2 - r_1^2}{r_0^2 - r_1^2} \end{bmatrix}$$

$$= \begin{bmatrix} r_0 \cdot \left(\frac{r_1}{r_0} - \frac{r_0 r_1 r_2 - r_1^3}{r_0(r_0^2 - r_1^2)} \right) + r_1 \cdot \left(\frac{r_0 r_2 - r_1^2}{r_0^2 - r_1^2} \right) \\ r_1 \cdot \left(\frac{r_1}{r_0} - \frac{r_0 r_1 r_2 - r_1^3}{r_0(r_0^2 - r_1^2)} \right) + r_0 \cdot \left(\frac{r_0 r_2 - r_1^2}{r_0^2 - r_1^2} \right) \end{bmatrix}$$

$$= \begin{bmatrix} r_1 - \frac{r_0 r_1 r_2 - r_1^3}{r_0^2 - r_1^2} + \frac{r_0 r_1 r_2 - r_1^3}{r_0^2 - r_1^2} \\ \frac{r_1^2}{r_0} - \frac{r_0 r_1^2 r_2 - r_1^4}{r_0(r_0^2 - r_1^2)} + \frac{r_0^2 r_2 - r_1^2 r_0}{r_0^2 - r_1^2} \end{bmatrix}$$

$$= \begin{bmatrix} r_1 \\ \frac{r_1^2(r_0^2 - r_1^2) - (r_0 r_1^2 r_2 - r_1^4) + r_0(r_0^2 r_2 - r_1^2 r_0)}{(r_0^2 - r_1^2) r_0} \end{bmatrix}$$

$$= \begin{bmatrix} r_1 \\ \frac{r_0^2 r_1^2 - r_1^4 + r_1^4 - r_0 r_1^2 r_2 + r_0^3 r_2 - r_0^2 r_1^2}{r_0(r_0^2 - r_1^2)} \end{bmatrix}$$

$$= \begin{bmatrix} r_1 \\ \frac{r_0 r_2(-r_1^2 + r_0^2)}{r_0(r_0^2 - r_1^2)} \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} = \text{RHS (Proved)}$$

Lecture-15

31/08/21

Roll no-21410103

② Covariance method →

In this method, we fix the interval over which the MSE is computed, and then consider the computation of $\phi_n(i,k)$ i.e.

$$E_n = \sum_{m=0}^{N-1} e_n^2(m)$$

$$\phi_n(i,k) = \sum_{m=0}^{N-1} s_n(m-i)\, s_n(m-k) \qquad 1 \le i \le p, \quad 1 \le k \le p$$

Now if we change the index of summation, we get

$$\phi_n(i,k) = \sum_{m=0-i}^{N-1-i} s_n(m)\, s_n(m+i-k)$$

This will span beyond the interval of consideration. We are not putting any constraints in the signal in covariance method. We need some more samples prior to the interval. Also,

$$\phi_n(i,k) = \sum_{m=-k}^{N-1-k} s_n(m) \cdot s_n(m+k-i)$$

We require samples from $-p \le m \le N-1$

The limits of the summation are not the same now. So, to evaluate $\phi_n(i,k)$, we need sample values from $-p$ to $N-1$ i.e. $-p \le m \le N-1$ i.e. beyond the interval under consideration. In this case no window is used.

We now get,

$$\sum_{k=1}^{p} a_k \phi_n(i,k) = \phi_n(i,0), \quad i = 1,2,\ldots,p$$

In the matrix form, we get

$$\begin{bmatrix} \phi_n(1,1) & \phi_n(1,2) & \cdots & \phi_n(1,p) \\ \phi_n(2,1) & \phi_n(2,2) & \cdots & \phi_n(2,p) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_n(p,1) & \phi_n(p,2) & \cdots & \phi_n(p,p) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_p \end{bmatrix} = \begin{bmatrix} \phi_n(1,0) \\ \phi_n(2,0) \\ \vdots \\ \phi_n(p,0) \end{bmatrix}$$

This cofactor matrix symmetric i.e. $\phi_n(i,j) = \phi_n(j,i)$ $\forall i,j$

(7)

$\phi_n(1,1)$, $\phi_n(2,2)$, ..., $\phi_n(p,p)$ are not same any more. They don't represent energy also.

But $\phi_n(i,k) \neq \phi(i+j, k+j)$

But it is not a toeplitz matrix any more.

We have to find the determinant and make sure it is not zero.

One good way of solving this is Cholesky decomposition method.

$$\underset{p\times p}{\phi}\ \underset{p\times 1}{\alpha} = \underset{p\times 1}{\psi}$$

Let $\phi = VDV'$ i.e. a product of 3 matrices

D is a diagonal matrix i.e. all the non-diagonal elements are zero.

V is a lower triangular matrix. In this matrix, all the diagonal elements should be 1.

So $VDV'\alpha = \psi$

Let $DV'\alpha = y$

So $\underset{p\times p}{V}\ \underset{p\times 1}{y} = \underset{p\times 1}{\psi}$

Solving this equation becomes very easy since V is a lower triangular matrix.

Analysis →

		Durbin's algorithm	Cholesky decomposition
①	Storage. Samples	N	$N+p$
②	Matrix	$\propto p$	$\propto p^2$
③	Window	N	0
④	Multiplication		
④	Windowing	N	0
⑤	Correlation	$\propto N \cdot p$	$\propto N \cdot p$
⑥	Matrix solution	$\propto p^2$	$\propto p^3$

(3)

Hamming window →

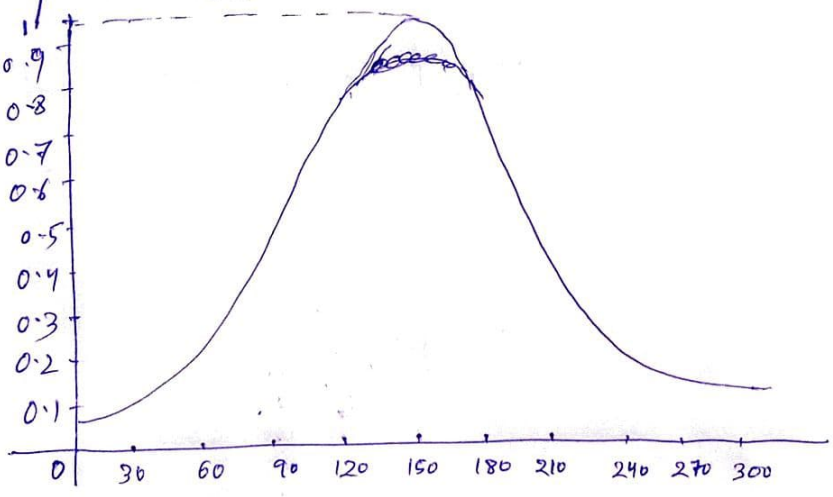
$$w(m) = 0.54 - 0.46 \cos\left\{\frac{2\pi m}{N-1}\right\} \quad ; 0 \le m \le N-1.$$

$$= 0 \quad ; \text{ otherwise}$$

MATLAB code to plot the formula →

```
1.  clear all;
2.  close all;
3.  clc;
4.  x = zeros(1,1);
5.  for index = 1:320
6.      x(1,index) = index-1;
7.  end
8.  y = 0.54 - 0.46*cos((2*3.14/319)*x);
9.  figure(1);
10. plot(x,y);
11. set(gca, 'XTick', 0:30:319);
12. xlabel('N');
13. ylabel('Hamming window');
14. title('Hamming window plot');
```

The plot will look like -



Lecture-16

06/09/21

Roll no- 214101037

When we try to regenerate the waveforms, we do not use a_i's directly. We invert the signs to get the final a_i's. We can use them to predict the next samples. We can use these a_i's to represent the waveform.
We assume that the signal is steady / stationary during the 20 ms where if the vocal tract has any changes then a_i's will change and we can compare across different signals for those 20 ms.

Cepstral coefficients →

LP coefficients will be converted to cepstral coefficients.
$C(m)$ is cepstral coefficient.

$c_0 = \log \sigma^2$; σ is a gain term of the LPC model i.e. $[R[0]]$.

$$c_m = a_m + \sum_{k=1}^{m-1} \left(\frac{k}{m}\right) c_k a_{m-k} \quad ; 1 \le m \le p$$

$$= \sum_{k=m-p}^{m-1} \left(\frac{k}{m}\right) c_k a_{m-k} \quad ; \text{where } m > p$$

C_i ; $i = 1, 2, 3, \ldots, Q$ and $Q \simeq \frac{3}{2}p$ & $p = 12$
$c_1, c_2, \ldots, c_m$ forms the feature vector.
c_0 forms the energy.
Q can also be equal to 12 i.e. p.

Properties of cepstral coefficient →

① The lower order cepstral coefficients are sensitive to spectral slope. c_1, c_2, c_3, c_4 vary

(2) widely if the lower formant frequencies vary a lot, e.g. for dipthongs they will vary.

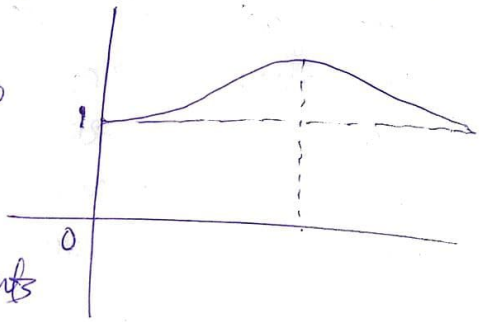
(2) The higher order cepstral coefficients are sensitive to noise or noise like sounds.

So, a tapered window is used to minimize these sensitivities.

The most popular weighing scheme used is

$$w(m) = \left[1 + \frac{Q}{2}\sin\left(\frac{\pi m}{Q}\right)\right] \quad ; 1 \le m \le Q$$

This is called raised sine window, which looks like the following.



The minimum weight is 1, so no information is lost.

So if we multiply w(m) to C(m), the middle coefficients will be amplified.

Inverse of filtering is called liftering.

The raised sine window is not filtering and liftering the C(m).

The C_i's with weights i.e. C_i's multiplied with W_i's form the features for that particular frame or window.

The cepstral representation of the speech spectrum provides a good representation of the local spectral properties of the signal for the given analysis frame. An improved representation can be obtained by extending the analysis to include information about the temporal cepstral derivative i.e. the slope. This is approximately given by (using orthogonal polynomial fit)

$$\frac{\partial}{\partial t} C_m(t) = \Delta C_m(t) \simeq \mu \sum_{K=K}^{K} K\, C_m(t+K)$$

Where μ is an appropriate normalization factor and $2K+1$ is the number of frames over which the computation is performed.

Usually $K=3$ and $\mu = 0.375$ are used.
We have got the C_i's. We can extend the analysis by taking 3 frames to left and right side each, along with the C_i's which are the static characteristics. Then we can find out the trend of change of C_i's. So we get the ΔC_i's also. Then we get an augmented vector with C_i's and ΔC_i's. So it gives some dynamic properties also. The slope of C_i's are also covered at that time.
Therefore speech for a segment is represented by augmenting the weighted Q cepstral coefficients with the Q temporal cepstral derivatives giving

$$P' = [C_1(t), C_2(t), \cdots, C_Q(t), \Delta C_1(t), \Delta C_2(t), \cdots, \Delta C_Q(t)]$$

It forms the complete feature vector.
For vowels we don't have to compute ΔC_i's.

Distance measures →

① Euclidean distance →

$$D_{cep} = \sum_{i=1}^{Q} \{C_i(t) - C_i(r)\}^2, \text{ where}$$

$t \rightarrow$ test data, $r \rightarrow$ reference data

It is simple to implement and gives equal weightage to each component of the vector.
The variability of C_i's are not taken into consideration. But the equal weightage is a demerit.
So if some C_i's are varying a lot, that will create problems.

② Mahalanobi's distance →

$$D_{Mcep} = (\tilde{C}_t - \tilde{C}_r)' \, V^{-1} \, (\tilde{C}_t - \tilde{C}_r);$$

V is the covariance matrix of the test feature vectors (C_i's)

$$(\tilde{C}_t - \tilde{C}_r)'_{1\times Q} \quad V^{-1}_{Q\times Q} \quad (\tilde{C}_t - \tilde{C}_r)_{Q\times 1}$$

(4)

Covariance tells us how the cepstral coefficients agree to each other.

Subtract the test cepstral coefficients and references, it gives a column vector. Take its transpose.

V is obtained from Durbin's algorithm, take its inverse and multiply with the column vector.

So if V is highly varying, V' will vary less.

Lecture - 17

07/09/21

Rollno - 214101037

Problems with Mahalanobi's distance →

① Although very accurate.
② V' computation is difficult
③ Many off diagonal elements are close to 0, we may end up getting errors.

This distance measure is theoretically very accurate but practically doesn't work well.

③ Tokhura's distance →

Tokhura suggested to take the diagonal elements only of V.

$$D_{Tcep} = \sum_{i=1}^{Q} w(i) \left[C_i(t) - C_i(r) \right]^2$$

where $w(i)$ is the inverse of the variance of ith element i.e.

$$w(i) = \frac{1}{\sigma_i^2}$$

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^2 , \quad \bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

We can take the cepstral coefficients which vary, the variance will be calculated and used in Tokhura's distance calculations.

Cepstrum →

The fourier transform of $x(n)$ is given by

$$X(K) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi}{N}Kn} \quad ; \quad 0 \le K \le N-1$$

Let $\hat{X}(K) = \log[X(K)]$

$$\hat{x}(n) = \frac{1}{N} \sum_{K=0}^{N-1} \hat{X}(K)\, e^{j\frac{2\pi}{N}Kn} \quad ; \quad 0 \le n \le N-1$$

We are taking log of the fourier transform and considering the log function as the fourier coefficient we are trying to regenerate the original signal.

But we won't get the original signal, we get something similar. Because log will change the properties of fourier coefficients.

It is the inverse fourier transform of the complex logarithm of the discrete fourier transform.

This has come back to time domain but it is not the original signal. We got the inverse of spectrum i.e. cepstrum.

Putting log on fourier coefficients restricts their movements. So we get a steady set of series.

Cepstrum has a one to one correspondence to cepstral coefficients produced from LPC. So we use them as the features of speech.

cepstrum is defined by

$$c(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \log |X(e^{jw})|\, e^{jwn}\, dw \quad ; \quad -\infty \le n \le \infty$$

or approximately $c(n) = \frac{1}{N} \sum_{K=0}^{N-1} \log|X(K)|\, e^{j\frac{2\pi}{N} \cdot Kn}$;

$0 \le n \le N-1$

If we take Tokhura's weight as 1, the distance will be Euclidean distance.

③

Distance measures →

④ Itakura minimum prediction residual distance →

$$D_{I_{LPC}} = \frac{a_r' R_t a_r}{a_t' R_t a_t} - 1$$

where a_r → LPCs of the reference data

a_t → LPCs of the test data

R_t → auto correlation matrix of the test data

$a_{r_{p\times 1}}$ $R_{t_{p\times p}}$

Denominator gives the mean squared error and it is supposed to be the minimum. Hence numerator is bigger than denominator and the ratio will be greater than 1. When reference data comes close to test data, ratio keeps on reducing.

Vowel recognition assignment →

Algorithm →

- → Given 320 samples.
- → Find out Ai, Ri, Ci and validate.
- → Hamming window and DC shift is not applied. Only normalization is done.
- → For checking purpose, no need to invert Ai's.

validation

- → 10 recordings for training, 10 for testing.
- → select 5 frames from the steady part.
- → Ri, Ai, Ci computation.
- → Here Ai's should be inverted.
- → Apply raised sine window to Ci's.
- → Above 5 steps should be repeated for 10 recordings.
- → 5 frames for each recordings gives 50 Ci's.
- → Frame k of all 10 recordings should be considered for average and hence 10 rows will produce 1 row of Ci's.
- → So finally 5 rows, 12 columns.

training

④ → Dump them in a text file. These are references.

→ We need to check, out of 10 recordings, how many are used correctly.
→ Take 5 stable frames.
→ Compute Ri, Ai, Ci.
→ Calculate Tokhura's distance from each reference file.
→ 5 frames for 5 corresponding rows. Take the average.
→ The one with minimum distance will be the answer.

testing

Pseudo code →

Validation

```
1. //normalization already done; if not then perform it
2. // Ri (from R0 to R12), read data from file to $arr
3. repeat $i = 0 to 12:
4.     repeat $j = 0 to 320-i-1:
5.         $r[i] = $arr[j] * $arr[i+j];
6.     end loop;
7. end loop;
8. //Ai (from A1 to A12), following Durbin's algorithm.
9. initialize $E[13], $K[13], $A[13][13], $a[12] to 0's;
10. $E[0] = $R[0];
11. repeat $i = 1 to 12:
12.     repeat $j = 1 to $i-1;
13.         $K[i] += $A[i-1][j] * $r[i-j];
14.     end loop;
15.     $K[i] = $r[i] - $K[i];
16.     $K[i] /= $E[i-1];
17.     $A[i][i] = $K[i];
18.     repeat $j = 1 to $i-1:
19.         $A[i][j] = $A[i-1][j] - ($K[i] * $A[i-1][i-j]);
20.     end loop;
21.     $E[i] = (1 - ($K[i] * $K[i])) * $E[i-1];
22. end loop;
23. //A[12][1] to A[12][12] are the final ai values.
```

```
repeat $i=1 to 12:
    $a[i-1] = $A[12][i];
end loop;
//Ci (from C0 to C12)
$C[0] = log($R[0]*$R[0]);
repeat $i=1 to 12:
    repeat $j=1 to $i-1;
        $C[i] += ($i/$j) * $C[j]*$a[i-j-1];
    end loop;
    $C[i] += $a[i-1];
end loop
//validate with given values.
```

Training

```
initialize $ctotal[10][5][12];
repeat $rec = 1 to 10:
    //applying DC shift
    initialize $count=0, $dc=0, $nf=0;
    repeat till EOF of $rec:
        $count+=1;
        $dc += value_read_from_$rec;
    end loop;
    $dc /= $count;
    //normalization
    initialize $max=0, $min=0;
    repeat till EOF of $rec:
        $x = read from $rec;
        $max = max($x, $max);
        $min = min($x, $min);
    end loop;
    $min = abs($min);
    $nf = 5000/(($max+$min)/2);
    truncate $nf upto 1 decimal point;
    repeat till EOF of $rec:
        $x = read from $rec;
        $x -= $dc; //dc shift
        $x *= $nf; //normalization
```

```
59.       end loop;
60.       read from the new data and take 5 steady frames;
61.       store them in $frames[]; compute ri, Ci, ai again
62.       repeat $i = 0 to 11:                    (follow line 1 to 34)
63.           repeat $f = 0 to 4:
64.               $ctotal[rec][f][i] = $C[i];
65.           end loop;
66.       end loop;
67.       //raised sine window
68.       repeat $i = 1 to 12:
69.           repeat $j = 0 to 4:
70.               $ctotal[rec][j][i-1] *= (1+(6*sin((3.14*$i)/12)));
71.           end loop;
72.       end loop;
73.   end loop; //processing done for 10 files
74.   //finding out Ci values to dump
75.   initialize $cfinal[5][12];
76.   repeat $i = 0 to 4:
77.       repeat $j = 0 to 9:
78.           repeat $K = 0 to 11:
79.               $cfinal[i][K] += $ctotal[j][i][K];
80.           end loop;
81.       end loop;
82.       repeat $K = 0 to 11:
83.           $cfinal /= 10;
84.       end loop;
85.   end loop;
86.   // dump $cfinal into a text file
87. //line no 36 to 86 should be repeated for all vowels.
88. // so we have five reference files, one for each vowel.
```

Testing

```
89.   // do similare steps from line no 36 to 61 for a file
90.   // you'll get 5X12 array for Ci's.
91.   // Tokhura's distance calculation
92.   initialize $w[12] with given weights;
93.   initialize $td[5] to 0;
```

```
94. repeat $vowel = 0 to 4:
95.     repeat $f = 0 to 4:
96.         repeat $i = 0 to 11:
97.             $td[vowel] = $w[i] * ($C[i] - $C_file[i]) * ($C[i] - $C_file[i])
98.         end loop;
99.     end loop;
100. end loop;
101. //minimum of $td will be the answer.
```

Lecture-18

10/09/2021

Roll no-21410a1037

(5) Itakura-Saito Distance →

$$Dis_{LPC} = \left[\frac{a_r' R_t a_r}{a_t' R_t a_t} - 1\right] + \log\left[\frac{G_t^2}{G_r^2}\right],$$

where $G = \sqrt{a' R a}$

We consider G as the gain term. After adding it, the Itakura distance measure is enhanced. Since it is log function, it slows highly jumping values down.

We can compare one waveform with another waveform. 20 ms of speech gives 320 samples. 16 bit signal with 10 kHz frequency will result in 160000 values per second.

Let p=10 in LPC, for 100 analysis/second, the amount of values will be 16000 values per second. (each c_i is of 16 bits)

This can be termed as compression i.e. using 16000 values in stead of 160000 values

The question is, can we do further compression. Voice signals are heavy but the same thing written using characters are light. So this

(2)

is a huge compression.

Converting speech to text is called "speech recognition". This brings the concept of vector quantization.

The concept of building a codebook of distinct analysis vectors (variations of each phoneme) is the basic idea behind vector quantization or VQ techniques.

The main goal here is to compress data. We require a codebook of about 1024 unique spectral vectors (25 variants of 40 basic speech sounds).

e.g.

$$\begin{array}{c} 1 \\ 2 \\ 3 \\ \vdots \\ 1024 \end{array} \begin{bmatrix} C_1 & C_2 & \cdots & C_{12} \\ C_1 & C_2 & \cdots & C_{12} \\ C_1 & C_2 & \cdots & C_{12} \\ \vdots & & & \\ C_1 & C_2 & \cdots & C_{12} \end{bmatrix}$$

This table is a codebook.

Then to represent an arbitrary vector, we need a 10-bit number ($2^{10} = 1024$) which is the index of the codebook that best matches the input vector. We calculate the C_i values of the speech, find out Tokhura's distance for all rows of codebook, take the minimum index. Then 10 bits are required to represent it. So we transfer such data through network. This is a huge compression.

If we have 100 vectors per second, then bit rate will be 1000 bits/second ($\because p = 10$). So the reduction is 16 fold i.e. 16000 to 1000.

Advantages →

① We need less storage space.

③

To handle vector quantization, only 10 bit numbers are needed. So it requires very little space.

② Reduced computation for determining similarity of cepstral coefficients/spectral vectors.
All we need is to compare two numbers. So speed increases.

③ Discrete representation of speech signals.
The text should come as an output which is discrete.

Disadvantages →

① Spectral distortion is introduced.
It means that there will be some quantization error. Because the size of codebook/vector quantizer is large. That is called quantization error which is always greater than 0.

② Storage of codebook vectors may be non-trivial for large sized codebook.
Large codebooks affect system performances.

Vector quantization →

Let $\vec{x}$ be a K-dimensional vector whose components are real valued random variables. A vector $\vec{x}$ is mapped on to another K-dimensional vector $\vec{y}$ and is written as

$$\vec{y} = q(\vec{x})$$

i.e. $\vec{y}$ takes on one of a finite set of values

$$Y = \{ \vec{y}_i \} \quad ; 1 \le i \le K$$

The set of vectors Y is called the codebook and each $\vec{y}_i$ is called a codeword, or template. The size K of the code-book is referred to as the number of levels.

- Gray (1984), Makhoul et. al (1985)

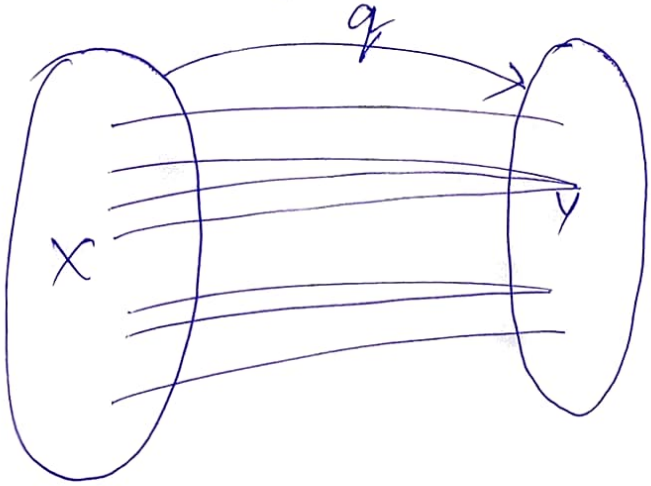
(4)

$\vec{x}$ will be 12-dimensional for $p=12$.

Random variables are measurements of phenomena.

K- dimension of $\vec{x}$

K- Number of entries in codebook



When $\vec{x}$ is quantized as $\vec{y}$, a quantization distortion measure $d(\vec{x}, \vec{y})$ can be defined between $\vec{x}$ and $\vec{y}$. The overall average distortion is then represented by

$$D = \lim_{M \to \infty} \frac{1}{M} \sum_{n=1}^{M} d(x_{(n)}, y_{(n)})$$

D tells us how good the vector quantizer is. If D is high, quantizer is not good and vice versa. It tells us the quality of the vector quantizer.

Lecture-19

13/09/21

Roll no- 214101037

There should be a family of vector quantizers i.e. the ycn) should be designed appropriately. Then the one with the minimum distortion(D) would be chosen for further processing.

A quantizer is said to be optimal (minimum distortion) if the overall distortion is minimized over all K-level quantizers.

To make such a codebook the K-dimensional space of $\vec{x}$ is partitioned into K regions i.e. $\{C_i\}$; $i=1,2,\cdots,K$ with $\vec{y_i}$ being associated with each region C_i. The quantizer then assigns the code vector $\vec{y_i}$ if $\vec{x_i}$ is in C_i i.e.

$$q(\vec{x}) = \vec{y_i}$$

Conditions necessary for optimality →

① Nearest neighbour selection rule →

$$q(\vec{x}) = \vec{y_i} \text{ iff } d(\vec{x},\vec{y_i}) \le d(\vec{x},\vec{y_j}) \quad \forall i \neq j \text{ and } 1 \le j \le K$$

It means quantize $\vec{x}$ to that $\vec{y_i}$ where the distance of $\vec{x}$ & $\vec{y_i}$ is the minimum among all $\vec{y_i}$'s. It means quantize $\vec{x}$ to $\vec{y_i}$ for that $\vec{y_i}$ for which the distortion is minimum.

② Centroid condition →

Centroid is the middle point of an object.

$\vec{y_i}$ should be chosen to minimize the average distortion in the region C_i. such ~~that~~ a vector is called the centroid of the region C_i if there are M_i vectors in the region C_i then the centroid is given by

$$\vec{y_i} = \frac{1}{M_i} \sum_{j=1}^{M_i} \vec{x_j} \quad ; \quad \vec{x_j} \in C_i$$

Generalized Lloyd's algorithm →

(K-means algorithm)

We shall use the two conditions of optimality along with this algorithm.

$\{x(n)\}$ is the universe of vectors.

Let K be the number of clusters. K regions

$\{C_i\}$; $i = 1, 2, \ldots K$

Step-1 → Initialization

Set $m = 0$ where m is the iterative index

Choose a set of initial code vectors $(y_i(0), 1 \le i \le K)$ using an adequate method.

There are K entries of K-dimensional vectors in the codebook.

Step-2 → Classification

We need to take all $\{x(n)\}$'s where $1 \le n \le M$ (large no.)

into the clusters i.e. $C_i(m)$. We call them as training sequence using the nearest neighbour selection rule.

We shall populate the clusters i.e. $\vec{x_i}$ will be put in the appropriate bucket.

$\vec{x} \in C_i(m)$ iff $d(\vec{x}, y_i(m)) \le d(\vec{x}, y_j(m)) \; \forall j \neq i$, $1 \le j \le K$

bucket here refers to codebook entries.

Step-3 → Code vector updation

set $m \leftarrow m+1$

Update the code vector of each cluster by computing the centroid of each cluster as $\vec{y_i}(m)$

$$\vec{y_i}(m) = \text{centroid}(C_i(m)) \; ; \; 1 \le i \le K$$

Then compute $D(m)$.

Step-4 → Termination

If $D(m)$ at iteration m ~~is~~ relative to $D(m-1)$ is below a certain threshold, stop,

else go to step-2.
That means do it till there is some improvement in the distortion.

The flaw in this algorithm lies at choosing an adequate method in step-1. This is also known as empty cell problem. How we choose the initial choice of code vectors, affects the entire process heavily.

If the code vectors are too much distributed, it may happen that some buckets will remain empty. It means the average distortion will go up. We can't compute centroid since Mi will be 0. This is empty cell problem. We should ensure that this never happens.

To solve this problem →

① Increase universe size →
Take a large collection of $\vec{x}$ vectors. So there will be no empty buckets, at least the probability is very low.

② Decrease size of K →
Try adjusting K to a reasonable number so that all variations of code vectors get covered.

③ Ensure the initial codebook to be a good one →
Here it is a deadlock. So, there will be many hit and trial iterations.

Lecture-20

14/09/21

Roll no - 21401037

To overcome the flaw of k-means, we have an enhancement.

Linde, Buzo and Gray algorithm →

This algorithm tries to improve the initial generation of codebook. It is also called the binary split algorithm or the modified k-means algorithm. The smallest possible codebook will be of size 1. Also, that will be the centroid or average or the arithmetic mean of the universe. This codebook will give the optimal output. Then we can gradually increase its size such that they remain optimal in nature.

Step-1 →

Design a one-vector-code-book. We have a long list of cepstral coefficients. We can take the average and that will be the code entry. This will be the centroid of the entire training vectors.

Step-2 →

Double the size of the codebook by splitting each entry as follows.

If the current entry is $\vec{y}_n$, then

$$\vec{y}_n^{+} = \vec{y}_n * (1+\vec{\varepsilon})$$

$$\vec{y}_n^{-} = \vec{y}_n (1-\vec{\varepsilon})$$

where n varies from 1 to the current size of the codebook and $\vec{\varepsilon}$ is the splitting parameter. $0.01 \leq \varepsilon \leq 0.05$.

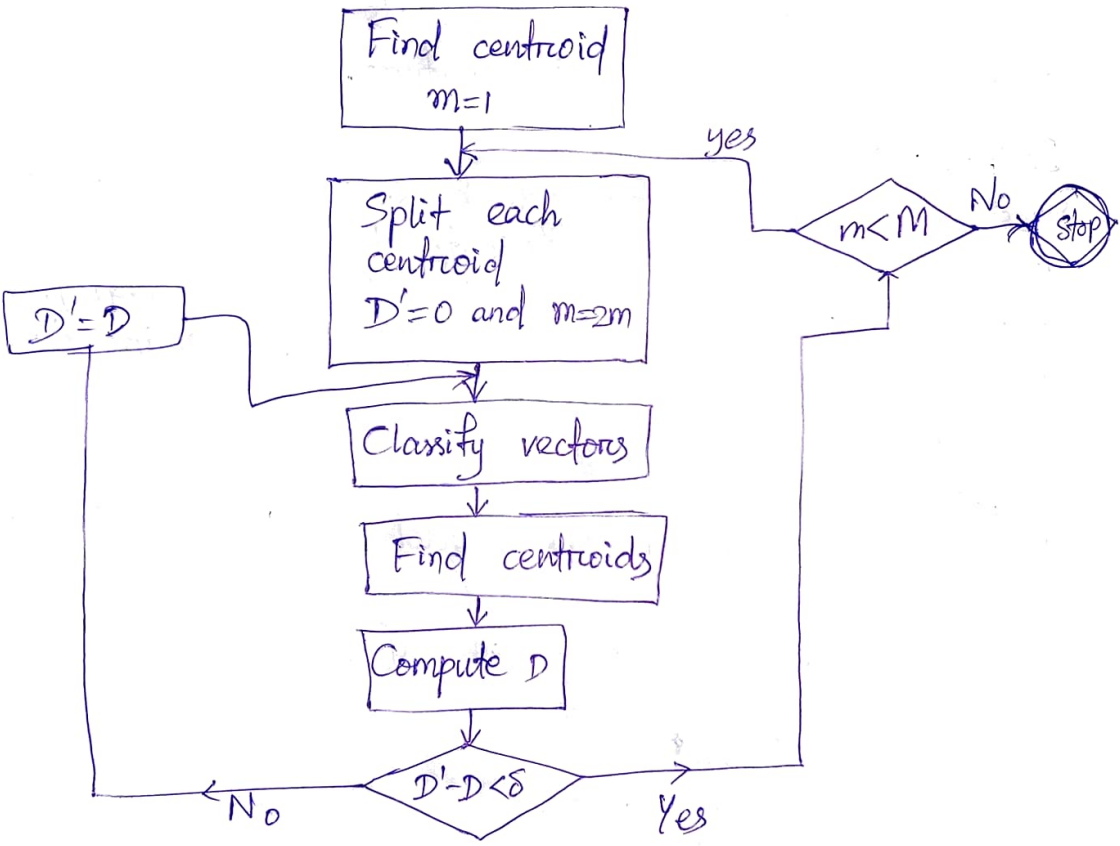
Now the codebook is double in size but not optimal.

Step-3 →

Use k-means algorithm to get the best set of centroids for the split codebook i.e. the

newly generated double sized codebook.
The entries will slowly stabilize and then we get the optimal codebook

Step-4→
Iterate steps 2 and 3 until a codebook of size M is reached.
Starting from codebook of size 1, splitting and getting a vector of double size, then using k-mea to make it optimal, repeat this process till the end.
The centroid of the universe becomes the input here. Selection of ε is important.



We get rid of the bias i.e. we don't need to choose the initial codebook.

Assignment→

K=8, K=12 (=p), 200 iterations (maximum), ε=0.03

① Generate a codebook of size K using k-means. Find out D. Initial codebook should be generated.

② Generate a codebook of size K using k-means.

Find out D. Here initial codebook should be the single centroid entry.
We need to find out the population of the buckets.
Compare the D of ① and ②.

Issues →

→ It may happen, splitting parameter will take the vector away from the universe. Then the cluster might get empty. This is empty cell problem and M_i becomes 0. This problem happens towards the end of the algorithm since the codebook has many entries and it is very much fragmented.

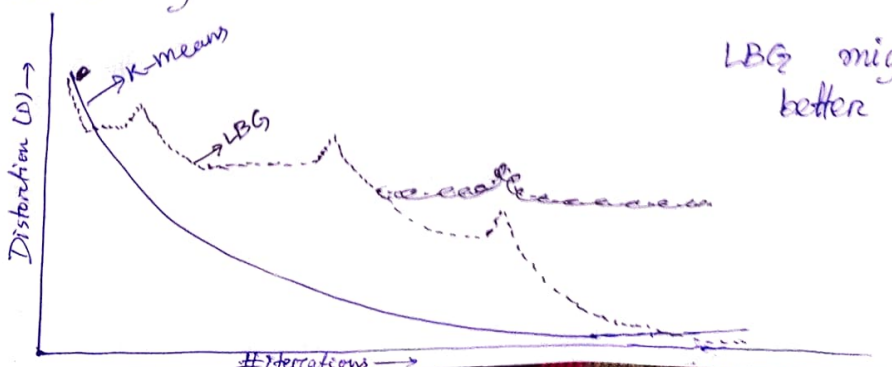
Solution →

→ Increase universe size. Data variety will be there.
→ Reduce codebook size.
→ Split the most dense cluster into two parts i.e. the entry with maximum number of $\vec{x}$s. Do this when the empty cell is reached. Run K-means again by throwing away the empty cell vector and putting the new vector. The distribution will be better and the solution will hence become better. We are splitting the dense entry to keep the codebook size same.

Extensions →

We can make multiple codebooks. Variety will be there. So we can have a hierarchy of codebook and the last one will give the actual result. Binary Search Tree can be used. There is a concept of matrix quantization also.

K-means algorithm vs LBG →



LBG might give better result.

Lecture - 21

17/09/21

Roll no - 21410103 7

Can we improve the codebook obtained by LBG?
Once we reach the desired size, we can start reverse engineering. We have optimal codebook and the vectors correspond to some speech sounds. If we can generate the sounds with those C_i's then we can find out which sounds are produced by LBG algorithm.
So we can study the nature of the sound and make another codebook with variations of the same sounds. So we shall have a tree whose root is the actual codebook with K entries and K children. Each child represents a single sound in a detailed manner. The complexity of the algorithm will go up. The qualitative improvement will be also higher. So, we can decide the codebook architecture based on the problem.
We can always have a hierarchy of codebooks. At destination, we need all the codebooks while decoding. This opens ways for security implementation. These codebooks can be used to encode and decode data to maintain security.
K-means algorithm is not providing any improvements to the codebook.
One way of improvement is using Chemistry. That means we can borrow the concept of "annealing". The process of glass making is called annealing. Normal glasses are very fragile but the tempered glasses or hardened glasses are not fragile. It doesn't break easily. These type of glasses are built using annealing. We allow the glass to cool down very slowly in this process. First we heat it to a very high temperature (molten state). There the glass has a huge amount of kinetic

energy. So when we cool it down slowly, the molecular gaps are highly reduced and the glass becomes really hard. The movements of the molecules are also highly reduced and the molecules are tightly packed.

We can use this concept here. When we have reached the optimal codebook, we introduce heat into the codebook. That means we generate random numbers to move the code vectors a bit. After that we run K-means again. There are a few chances that we shall get a better codebook. Basically the codebook moves out of the local minima and converges to a better minima. We can do this process a huge number of times to get better codebook. But this is a very expensive process. But if expenses permit, we can use "Simulated Annealing" to get an optimal codebook.

Note: Since simulated annealing concept appeared, we can actually use any non-linear optimization method like genetic algorithm, particle swarm optimization etc. Genetic algorithm gives even better results.

Lecture - 22 28/09/21

Roll no - 21410103 7

After vector quantization we get some numbers. We can regenerate the codebook from the numbers and signal from the codebook. So, the entire process is reversible.

So, the numbers we get after quantization can result in different result if changed. The properties of speech are different and hence the numbers will also differ. In that case we won't be able to match the signal and say if they represent the same speech or not.

This brings a new Topic i.e. Data modelling. The features generated from vector quantization (or observations) will be used to do data modelling. That model will absorb these variations and will keep a few information. This model will compare the speech and will give a percentage of match. It will give a score how close or how far the speech samples are.

Taking average or choosing any one will not work always, it might result in information loss.

Discrete Markov processes →

Consider a system which may be described to be in one of any distinct N states, $S_i (i = 1, 2, \cdots, N)$ e.g. Banking Software. Before 10 o'clock the state is different, between 10-2 the state is different, for online processes the state is different and so on. We humans have state of sleeping, eating, fighting etc. At regular intervals, the system undergoes a change of state according to a set of probabilities associated with the state. Time is given by;

$t = 1, 2, \ldots, \ldots$ and the actual state at time 't' is given by q_t. Change in state can be from q_t to q_t also.

$$P[q_t = S_j / q_{t-1} = S_i, q_{t-2} = S_K, \ldots]$$

This gives the conditional probability of $q_t = S_j$ given that $q_{t-1} = S_i$, $q_{t-2} = S_K$ and so on.

$$= P[q_t = S_j / q_{t-1} = S_i]$$

It means there is a memory loss. It says that it depends only on the state of previous time point. This is called Markov chain.

$= a_{ij}$ (probability of $(1 \le i, j \le n)$ being in S_j at t given that at $t-1$ it was at S_i)

▷ The Markovian first order chain says we have a memory loss.

Properties ⟶

① $0 \le a_{ij} \le 1$

② $\sum_{j=1}^{N} a_{ij} = 1$ where $1 \le i \le N$

e.g. Let there be three states

state 1 : raining
state 2 : cloudy
state 3 : sunny

At a particular time, the system will be in any one state. What is the probability, according to the given model that the weather for eight consecutive days will be - sunny, sunny, sunny, rain, rain, sunny, cloudy, sunny? Today it is sunny.

$$A = \{a_{ij}\} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

$P(O/\text{model}) = P(3, 3, 3, 1, 1, 3, 2, 3/\text{model})$

O ↓ (Observation sequence)

$$= P(3) * P(3|3) * P(3|3) * P(1|3) * P(1|1) * P(3|1) * P(2|3) * P(3|2)$$

$$= 1 * 0.8 * 0.8 * 0.1 * 0.4 * 0.3 * 0.2 * 0.2$$

$$= 1.536 * 10^{-4}$$

Lecture-23

01/10/21

Roll no - 21410103 7

Coin tossing experiment with ghost →
Go to a ghost house. Ghost will toss coin(s). It won't tell you which coin and how many coins are used. It will only tell you the results. Finding out the number of coins and material is tough.
→ We can find out the probabilities and it will explain the phenomena. This is for a single coin.
→ Problem occurs when we have multiple coins. Probability of choosing the coins also comes into picture.

Hidden Markov Model →

It is a doubly embedded stochastic process with an underlying stochastic process that is not observable. (i.e. hidden) It can be observed through another set of stochastic processes that produces the sequence of observations.

Elements of an HMM →

① N → Number of states ; $S = \{S_1, S_2, \ldots, S_N\}$
at time t, state is going to q_t i.e.
$t \rightarrow q_t$; $q_t = S_1$ or S_2 or ... or S_N

② M → Number of distinct observation symbols per state (i.e. codebook size)
In coin tossing, $M = 2$ (H & T).
In case of speech, M is the codebook size.

③ The state transition probability distribution

(2) $A = \{a_{ij}\}_{N \times N}$; $a_{ij} = P(q_{t+1} = S_j \mid q_t = S_i)$ where $1 \le i, j \le N$

(3) Observation symbol probability distribution in state S_j

$B = \{b_j(k)\}$; $1 \le j \le N$ and $1 \le k \le M$

where $b_j(k) = P(v_k$ at time $t \mid q_t = S_j)$. Matrix size is $N \times M$.

(5) Initial state distribution

$\Pi = \{\Pi_i\}$ where

$\Pi_i = P\{q_1 = S_i\}$; $i = 1, 2, \ldots, N$

i.e at time $t = 1$, state S_i

$\sum_{i=1}^{N} \Pi_i = 1$ because one state must be true i.e. the system will be at one out of the N states

Lecture - 24

02/10/21

Roll no - 214101037

In coin tossing experiment, coins are the states. So we don't know how many coins are there.

In $A = \{a_{ij}\}_{N \times N}$, the row sum gives 1 because we'll anyway go to one of the given states.

The matrix in which row sums are 1, is called stochastic matrix.

In $B = \{b_j(k)\}_{N \times m}$, the rowsum again gives 1 because it will give one outcome out of M.

So both A and B are stochastic matrices.

Π vector also gives 1 as sum.

Marbles and Jar experiment ⟶

Choose a jar. Choose a marble from the chosen jar. Note it down and put it back.

This is also a hidden model since we can see the observations only.

→ A matrix tells us, given we chose a jar what is the probability of choosing a particular jar next?

→ B matrix tells us, the distribution of colors within a particular jar.

→ π tells us the probability of choosing a particular jar before starting the process.

→ e.g. in A, a_{33} denotes we chose 3rd jar. Now what is the probability of choosing the 3rd jar again? In B, $b_4(K)$ denotes choosing a marble in 4th jar. In π, we can fix that we'll choose 1st jar always. This makes the experiment becomes biased.

→ We don't know the values but we'll estimate them. There are various algorithms for this.

We can write an HMM as $\lambda = (A, B, \pi)$.

Problem - 1 → Evaluation Problem →

Given a model λ and a sequence of observations, how to compute the probability that the observation sequence was produced from the model. Also, it is called the scoring problem. That means, how well a given model matches an observation sequence.

The solution to this problem can be used in real time.

There can be different models we need to find the probability of the observation sequence, given the model λ_i i.e. $\max_i P(Q/\lambda_i)$; $i = 1, 2, 3, \ldots, W$

where W is the size of the vocabulary.

Problem - 2 → Problem of uncovering

We try to uncover the hidden part of the model. The hidden part is the number of states and their sequence. So, we shall find

the correct state sequence. We try to use some optimality criteria to solve this problem as best as possible.

Problem - 3 → Re-estimation problem →

We try to optimize the model parameters to best describe, how a given observation sequence comes about. The observation sequence used to adjust the model parameters is called the training sequence.

Lecture - 25

05/10/21

Roll no - 214101037

Consider a simple HMM-based speech/word recognition system having w words. (The model can recognize words given a vocabulary).
For digit recognition system $w = 10$ i.e. there will be 10 digits.
We want to generate a separate HMM for each of the w words.
Consider N states are there in each HMM.
Speech is represented by spectral vectors.
They are further encoded using vector quantization.
Here M = codebook size i.e. K
These are the observations.
So, we get a sequence of code vectors (actually the indices of the vectors in the codebook).
Let M be the size of the codebook.
So we will get the sequence of vectors out of M different indices.
So, training sequence will consist of one or more word by one or more speakers. If we ~~accure~~ utter a word 5 times, all of them will be the part of the training sequence.
The first task is to build the word model for each word. We'll use the solution of

problem-3 i.e. adjusting the model parameters. It will help us in building the Hmm.

To understand what this model physically means in terms of states etc. we use the solution to problem-2 to segment each of the word training sequence into states and study the properties of the spectral vectors that led to the observations occuring in each state. The goal here is to refine the model (more states or bigger codebook). This basically tries to adjust the hidden part of the model.

So, problem-3 gives a model. To check if it is good, problem-2 is used.

Finally, once we have w HMMs, recognition of an uttered word (unknown) is done using the solution to problem-1 to score against each word model based upon the given ~~train~~ test observation sequence and select the word whose model score is the highest.

The test sequence is fixed. We are going to take one HMM at a time and solve problem-1 and get a probability. After doing this w time, we choose the highest one.

Development of HMMs using problem-3,
Understanding it using problem-2,
Recognition of word using problem-1.

Problem-1 is required in real time.

We don't use the problems, we use their solutions.

Solution to problem-1 →

We want to calculate probability (O/λ).

$O_1, O_2, \dots, O_T$ ⟶ observation sequence

We can enumerate all possible state sequences of length T.

$N \rightarrow$ no of states, $T \rightarrow$ no of observations, then N^T

state sequences can be generated.
One of them would be $q = (q_1, q_2, q_3, \cdots, q_T)$
where q_1 is the initial state.
Now, the probability of O given the above state sequence is

$$P(O/q, \lambda) = \prod_{t=1}^{T} P(O_t/q_t, \lambda)$$

We assume that O_i's are independent, being in state q_1
↳ probability of throwing 1st observation
Therefore $P(O/q, \lambda) = b_{q_1}(O_1) \cdot b_{q_2}(O_2) \cdots b_{q_T}(O_T)$
Now, the probability of such a sequence q is given by

$$P(q/\lambda) = \pi_{q_1} \cdot a_{q_1 q_2} \cdot a_{q_2 q_3} \cdot \cdots \cdot a_{q_{T-1} q_T}$$

The joint probability of O and q is given by

$$P(O, q/\lambda) = P(O/q, \lambda) \cdot P(q/\lambda)$$
$$= \pi_{q_1} b_{q_1}(O_1) \cdot a_{q_1 q_2} \cdot b_{q_2}(O_2) \cdot a_{q_2 q_3} \cdot \cdots \cdot b_{q_{T-1}}(O_{T-1}) \cdot a_{q_{T-1} q_T} \cdot b_{q_T}(O_T).$$

Finally, $P(O/\lambda)$ is obtained by summing this joint probability over all possible state sequences q, giving

$$P(O/\lambda) = \sum_{\text{all } q} P(O/q, \lambda) * P(q/\lambda) \quad \text{——— } (*)$$

Lecture - 26

08/10/21

Roll no - 21410103 7

$N \rightarrow$ No of states
$T \rightarrow$ No of observations

Interpretation →

At time $t=1$, we are in state q_i with probability π_{q_i} (initial state probability) and generate the symbol O_1 with probability $b_{q_i}(O_1)$. We'll get values from λ.
At time $t=2$, we make a transition from state q_i to q_j with probability $a_{q_i q_j}$ and generate the symbol O_2 with probability $b_{q_j}(O_2)$.
This process continues until we make the last transition from q_{T-1} to q_T with probability $a_{q_{T-1} q_T}$ and generate the symbol O_T with probability $b_{q_T}(O_T)$.
The above equation requires order of $2T$ computations for each series and there are N^T sequences. So $2T * N^T$.
There are N possible states that can be reached $(N^T)^{\text{sequences}}$ and for each state sequence we have about $2T$ calculations for each term in the sum of equation - *.
Consider the situation $N = 5$ and $T = 100$

Then $2T * N^T = 1.578 * 10^{72}$ (huge!)

This is not feasible for a real time implementation.
There is an algorithm, which gives identical answer by exploiting the structure of the data. This will ensure that we have a real time implementation.
This procedure computes $P(O/\lambda)$ efficiently. The time complexity will be quadratic. This is orders of magnitude reduction.

There are two procedures. They are -

② The forward procedure →

Consider a forward variable $\alpha_t(i)$ as follows.

$\alpha_t(i) = P(O_1 O_2 \cdots O_t, q_t = S_i / \lambda)$

i.e. probability of partial observation sequence. i.e. $O_1 O_2 \cdots O_t$ until time t and state S_i at time t given the model λ.

We can solve for $\alpha_t(i)$ inductively as follows. We don't care about the past states, we only consider time point t i.e. at t the state is S_i, given the model λ.

Step-1 — Initialization →

$$\alpha_1(i) = \pi_i b_i(O_1) \quad ; \quad i = 1, 2, \cdots, N$$

Step-2 — Induction →

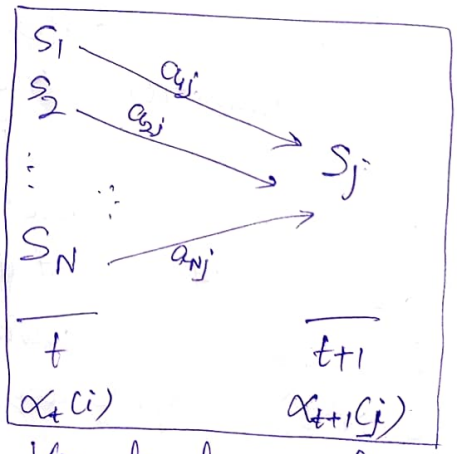
$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) \cdot a_{ij}\right] \cdot b_j(O_{t+1}) \quad ; \quad 1 \le t \le T-1, \; 1 \le j \le N$$

$\alpha_{t+1}(j)$ is the next time point. $\alpha_t(i)$ is the partial observation sequence at current time point. The next state should be j, so we need to transit from i to j by multiplying a_{ij}. Since we don't care about the current state i, so we sum them up over i. After reaching at j, we need to multiply $b_j(O_{t+1})$ to produce the output symbol O_{t+1}.

Step-3 — Termination →

$$P(O/\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

$O(N^2 T)$ computations are done in step-2. In step-3 we are still in S_i state with $\alpha_T(i)$ i.e. probability of partial observation sequence $O_1, O_2, \ldots, O_T$. Since we don't care about i, we sum them



Visualization of step-2 (This figure shows

(3) up to get the final probability.
For $N=5$, $T=100$
$N^2T = 2500$ operations
10^{72} to 2500 is orders of magnitude reduction.
2500 is the number for a single word. So, if we have w words it will be $2500 * w$.

how state S_j can be reached at time $t+1$ from N possible states.)

The backward procedure →

Consider a backward variable $\beta_t(i)$ as follows.
$\beta_t(i) = P(O_{t+1} O_{t+2} \cdots O_T / q_t = S_i, \lambda)$
At time t the state was S_i. We are looking at the future part. So, it is the probability of $O_{t+1} O_{t+2} \cdots O_T$ given that $q_t = S_i$ & λ.

Step-1 — Initialization →
$\beta_T(i) = 1 \; ; \; 1 \le i \le N$

Step-2 — Induction →
$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} \, b_j(O_{t+1}) \cdot \beta_{t+1}(j) \; ; \; 1 \le t \le T-1$$

The initialization step arbitrarily defines $\beta_T(i) = 1$ for all i.
Step-2 shows that in order to be in state i at time t and to account for the observation sequence from t+1 onwards you have to consider all possible states j at time t+1 accounting for the transition from i to j (a_{ij}) as well as the observation O_{t+1} in the state j ($b_j(O_{t+1})$) and then account for the remaining partial observation sequence from state j i.e. $\beta_{t+1}(j)$. This β will be used in solving problem-2. β also requires $O(N^2T)$ operations.

Lecture-27

12/10/21

Roll no - 214101037

The outcome sequence (HTTHT... in case of coin tossing experiment) corresponds to the sequence of indices of the codebook entries, i.e. the vector quantized output of a given speech utterance. Given there is a λ, we can now use the forward procedure.

Solution to problem-2 →

There are several possible ways of solving problem-2, i.e. finding out the optimal state sequence associated with the given observation sequence (i.e. which coin was tossed in case of a coin tossing experiment). The problem is to define what is optimal. One solution can be to choose the states q_t which are individually most likely. This leads to maximization of the expected number of correct individual states. To implement this solution, we define a variable $\gamma_t(i)$ as follows.

$$\gamma_t(i) = P(q_t = S_i \mid \underset{\sim}{O}, \lambda)$$

i.e. the probability of being in state S_i at time t given the observation sequence and the model λ. Using the definition of forward and backward procedures $\gamma_t(i)$ can be written as

$$\gamma_t(i) = \frac{P(q_t = S_i, \underset{\sim}{O} \mid \lambda)}{P(\underset{\sim}{O} \mid \lambda)}$$

$$= \frac{\alpha_t(i)\,\beta_t(i)}{P(\underset{\sim}{O} \mid \lambda)}$$

$$= \frac{\alpha_t(i)\,\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\,\beta_t(i)}; \quad t = 1, 2, \ldots, T$$

The Q in step-1 can be divided into two parts $\alpha_t(i)$ and $\beta_t(i)$ since $\alpha_t(i)$ indicates the past and $\beta_t(i)$ ~~and~~ indicates future. i means the current state is S_i. This division is done in step-2.

$P(Q|\lambda)$ means the probability of the entire observation sequence, given λ. We don't care about the state. Hence, we do sum over i in the denominator. This is done in step-3.

$\alpha_t(i)$ accounts for the partial observation sequence $O_1, O_2, \dots, O_t$ and state S_i at time t while $\beta_t(i)$ accounts for $O_{t+1}, O_{t+2}, \dots, O_T$ given state S_i at time t.

The normalization factor makes $\gamma_t(i)$ into a probability measure, so that

$$\sum_{i=1}^{N} \gamma_t(i) = 1$$

Since the denominator is already independent of i, this above expression will sum the numerators up, and then it will get cancelled out, so the result is 1.

Using $\gamma_t(i)$ we can solve for the individually most likely state q_t at time t as

$$q_t = \underset{1 \le i \le N}{\arg\max} \left[\gamma_t(i) \right], \quad 1 \le t \le T$$

The last equation maximizes the expected number of states i.e. choosing the most likely state for each time point t.

There can be some problems with the resulting state sequence e.g. if some state transition probabilities i.e. a_{ij}'s are 0's for some i and j then the optimal state sequence may not be a valid one.

This is because we consider only the most likely state at every time instant without giving regard to the probability of occurrence of a sequence of states.

One possible solution is to modify the optimality criterion e.g. →

Solve for state sequence that maximizes the expected number of correct pairs of states $(q_t q_{t+1})$ or triples of states $(q_t q_{t+1} q_{t+2})$.

These are reasonable for some applications but the most widely used criterion is to find the single best state sequence (path) i.e. maximize $P(\underset{\sim}{q} | \underset{\sim}{O}, \lambda)$ which is equivalent to maximizing $P(\underset{\sim}{q}, \underset{\sim}{O} | \lambda)$.

A formal technique exists and is based on dynamic programming method. It is called the Viterbi algorithm.

Lecture - 28

18/10/21

Roll no - 21410037

Viterbi Algorithm →

To find the single best state sequence $Q = \{q_1, q_2, \dots, q_T\}$ for a given observation sequence $O = \{O_1, O_2, \dots, O_T\}$ we need to define the quantity

$$\delta_t(i) = \max_{q_1 q_2 \dots q_{t-1}} P[q_1, q_2, \dots, q_t = S_i, O_1, O_2, \dots, O_t | \lambda]$$

i.e. $\delta_t(i)$ is the best score (highest probability) along a single path, at time t which accounts for the first t observations and ends in state S_i.

By induction we have

$$\delta_{t+1}(j) = \max_i [\delta_t(i)\, a_{ij}]\, b_j(O_{t+1})$$

To actually retrieve the state sequence we need to keep track of the argument that maximized $\delta_{t+1}(j)$ (i.e i) for each t and j. We do this by using $\psi_t(j)$.

The complete procedure for finding the best state sequence can now be stated as -

Step-1 - Initialization →

$$\left.\begin{aligned} \delta_1(i) &= \pi_i\, b_i(O_1) \\ \psi_1(i) &= 0 \end{aligned}\right\} \quad 1 \le i \le N$$

Step-2 - Recursion →

$$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i)\, a_{ij}]\, b_j(O_t)$$

$$\psi_t(j) = \operatorname{argmax}_{1 \le i \le N} [\delta_{t-1}(i)\, a_{ij}]$$

for $2 \le t \le T$

$1 \le j \le N$

Step-3 – Termination →

$$p^* = \max_{1 \le i \le N} \left[\delta_T(i) \right]$$

$$q_T^* = \operatorname{argmax}_{1 \le i \le N} \left[\delta_T(i) \right]$$

Step-4 – State sequence (path) backtracking →

$$q_t^* = \psi_{t+1}\left(q_{t+1}^*\right), \quad t = T-1, T-2, \cdots, 1$$

This is similare to the forward procedure (except for the maximization step). Here we maximize.

p^* tells us how good the state sequence is helping in modelling.

Lecture - 29

22/10/21

Roll no - 21410103 7

Inputs to Viterbi algorithm →
① The observation sequence
② λ (the model)

Goal is to associate states to the observations.

q_T^* is the most optimal state sequence. But it doesn't give us any hint, how to use it properly.

a matrix is not fully populated.

Dimension of B will be 5×32 for the assignment.

Codebook size is 32.

Solution to Problem-3 →

We want to determine a set of parameters, $\lambda = (A, B, \Pi)$ to maximize the probability of occurrence of the observation sequence given the model.

There is no known way to do this.

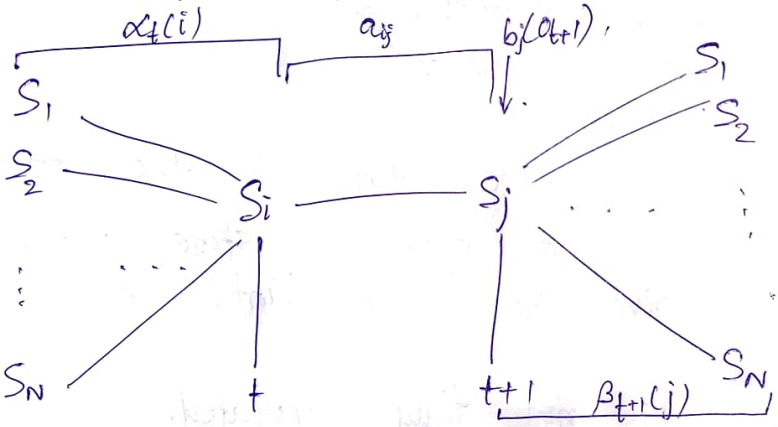
We can choose a set $\lambda = (A, B, \Pi)$ such that

$P(Q|\lambda)$ is locally maximized.
We use the Baumwelch method or equivalently the EM (Expectation Modification) method or gradient techniques.

To solve this problem using iterative update and improvement of HMM parameters, we define a variable $\xi_t(i,j)$ i.e. the probability of being in state S_i at time t and state S_j at time $t+1$, given the model λ and the observation sequence.

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j | Q, \lambda)$$

We have to transit to state S_j from any state at time t and then we have to go to S_j at $t+1$, then again we can go to any possible state. That means -



Using the definition of forward and backward probabilities we can write $\xi_t(i,j)$ as

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij}\, b_j(O_{t+1}) \cdot \beta_{t+1}(j)}{P(Q|\lambda)}$$

$$= \frac{\alpha_t(i)\, a_{ij}\, b_{t+1}(j) \cdot \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i)\, a_{ij}\, b_{t+1}(j)\, \beta_{t+1}(j)}$$

The denominator makes $\xi_t(i,j)$ into a probability

measure.

$\gamma_t(i)$ = probability of observation sequence ending in state S_i at time t given λ

We can now relate $\gamma_t(i)$ and $\xi_t(i,j)$ as

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$$

If we sum ~~over~~ $\gamma_t(i)$ over t, time index from 1 to T-1, we get a quantity which can be interpreted as the expected number of times the state S_i is visited or expected number of times transitions are made from state S_i.

Similarly, summation of $\xi_t(i,j)$ over t from 1 to T-1 can be interpreted as expected number of transitions from state S_i to S_j i.e.

$\sum_{t=1}^{T-1} \gamma_t(i)$ = expected number of transitions from S_i

$\sum_{t=1}^{T-1} \xi_t(i,j)$ = expected number of transitions from S_i to S_j

Using these formulae we give a reasonable set of reestimation formulae for π, A and B as follows.

$\bar{\pi}_i$ = expected frequency in state S_i at time $t=1$

$= \gamma_1(i)$; $i = 1, 2, \ldots, N$

$$\bar{a}_{ij} = \frac{\text{expected no of transitions from } S_i \text{ to } S_j}{\text{expected no of transitions from } S_i}$$

$$= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

(4)

$$\overline{b_j(k)} = \frac{\text{expected no of times in } S_j \text{ and observing symbol } v_k}{\text{expected no of times in } S_j}$$

$$= \frac{\sum_{t=1}^{T} \text{such that } O_t = v_k \; \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

If we define

Lecture - 30

25/10/21

Roll no - 214101037

If we define $\lambda = (A, B, \pi)$ as the current model and compute new values $\bar{\lambda}$ by using the above formula(e) it is proved that either -

① λ defines a critical point of the likelihood function in which case λ will be equal to $\bar{\lambda}$.

or

② $\bar{\lambda}$ is more likely than model λ in the sense that $P(O/\bar{\lambda}) > P(O/\lambda)$.

i.e. we have found a new model $\bar{\lambda}$ from which the observation sequence is more likely to have been produced.

But if $P(O/\bar{\lambda}) < P(O/\lambda)$ then we should throw away the new model.

Point - ① says that no improvement is possible. We have reached a critical point.

T should not exceed 150, otherwise overflow will occur and the program will crash.

$\pi = \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}$, $N=5$, $M=32$

$$A = \begin{bmatrix} \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5} \\ \vdots \end{bmatrix}_{5\times5}$$

(2) $B = \begin{bmatrix} \frac{1}{32} & \frac{1}{32} & \cdots \\ \vdots & & \end{bmatrix}_{5\times32}$ → unbiased HMM

Since we can transit from anywhere to anywhere, this is a complete and unbiased model. But, in reality, speech doesn't behave that way. Speech doesn't need such a complicated and unbiased model.

If all states are reachable from any state, then the model is called an Ergodic model.

When we start with a new model, we don't have any idea about it.

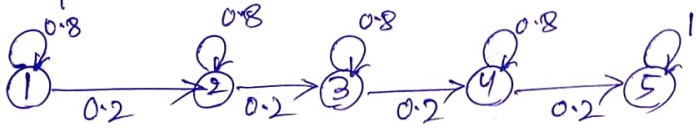
$$\Pi = [1, 0, 0, 0, 0]$$

$$A = \begin{bmatrix} 0.8 & 0.2 & 0 & 0 & 0 \\ 0 & 0.8 & 0.2 & 0 & 0 \\ 0 & 0 & 0.8 & 0.2 & 0 \\ 0 & 0 & 0 & 0.8 & 0.2 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}_{5\times5}$$

$$B = \begin{bmatrix} \frac{1}{32} & \frac{1}{32} & \cdots \\ \vdots & & \end{bmatrix}_{5\times32}$$

This model has constraints and it is powerful in case of speech.

Π says that we have to start at state-1 definitely. We can transit to other states later but we have to keep the starting point fixed.



A says we have a fixed direction of movement. We can't go in reverse direction. So there is a restriction on starting point.

(3)

There is a restriction on movement also. The rowsum must be 1 so that the matrices become stochastic.
This model is called a Bakis model or a feed forward model.
Movement of single step is allowed in this model.
When we utter anything, we get a sequence like 8 8 8 3 3 8 8 7 7 15 15 15 12 ....
This is the observation sequence.

1→ Call Viterbi algorithm. Get P^* and q^*_T.
P^* tells the quality of model (which is really bad for speech because of unbiased data).
q^* goes like 1 1 2 2 2 3 3 4 5 5 5 5 5 ... (T times).

2→ $\alpha, \beta, \gamma, \psi, \xi$ should be global to avoid overloading the program.

3→ A, B have to be recomputed.

Lecture-31

26/10/21

Roll no- 21410103 7

→ Since P^* value was very small, we need to use different format specifiers like %g, %e etc.

→ We need to run our respective programs and share the results.

→ Observation sequence is the indices of the codebook. We need to find out the indices of the cepstral coefficients. It will be the closest match of the input speech samples.

Output of Viterbi Algorithm →

We shall start with the feed forward model, then we shall use Viterbi Algorithm to find out P^*. Then we need to print the state sequence exactly below the observation sequence. It should have an one to one correspondence.

① Speech sample → Front end analysis (Ai, Bi, Ci ...) → Ci → LBG → observation seq.

② O.S. → Viterbi →

How to improve λ →

→ Run Viterbi algorithm to get P^* (quality of the model) and q_T^*. Print one below other (state and observation sequences). Use the solution to problem-3 to update the λ or to get a fresh $\bar{\lambda}$. π will remain same but A and B will get updated.

→ Use this $\bar{\lambda}$ and run Viterbi algorithm again.

→ Then we need to check whether the new p^* is better or not. p^* should be larger than the previous one (e.g. e-250 to e-150) to be tagged as "better".

→ In this case we can overwrite the λ with $\bar{\lambda}$ (i.e. A and B matrix).

→ This process should be repeated till P^* is ~~not~~ improving. Once it converges to values like e^{-18} or e^{-15}, we can stop the process.

→ Then we need to report the final λ on screen.

This is the flowchart. ↓

→ Hard code to initialize data structure to 0.

→ For each utterance we shall get a model.

→ If there are 20 utterances, there will be 20 models. They will look very different from each other. Then we shall compress 20 λ's to a single λ. That means we need to take the average for Π, A and B.

→ We need to verify whether the resultant matrices are stochastic or not i.e. we need to check if the row sum is 1. If it is not 1 then we need to make it 1 by finding out what is missing.

→ For this, we need to add the two values of one row in A and get the difference of it from 1. Then we need to update the larger value.

e.g. → if the two values are 0.75 and 0.2

3

we need to make it 0.8 and 0.2. Similarly if it is 0.85 and 0.2, we need to make it 0.8 and 0.2. This will make A stochastic.

→ The resultant λ contains the generic behaviour of all utterances of the same digit/word.

→ This average λ will be the starting λ. Now if we try to improve it using all previous steps, the new λ will be far superior.

→ If we do it 20 times (for 20 utterances), we shall get 20 new models. Again we can average it out and do the process again.

→ 3-4 rounds of averaging out will give a very superior value of P^*.

This whole process should be done for all ten digits/words.

Issues after generating λ →

→ Many values in B matrix will be 0. In B matrix, some observations which are in state 1 will be non-zero whereas others will be 0. The 0 values will never become non-zero, it might create problems during averaging.

Solution → if the entry is 0 in B, change it to $1 * 10e^{-30}$.

The matrix won't remain stochastic any more. The row sum will be more than 1. Subtract some values from the highest value in that row, to make it stochastic.

4

Layout of remaining part of course →

→ HMM, Dynamic Time Warping, Linear Time Warping

→ There are existing source codes, we can use.

→ The solution to these three problems will be used in final project.

→ A few no. of problems will be posted. We need to implement a voice control based solution for it (any one problem).

→ All the programs written till now will be used in the final project.

→ This model can be used for applications like language learning, rehabilitation etc.

→ Evaluation will be by 18th November.

Lecture - 32

29/10/21

Roll no - 21401037

Advantages of HMMs →

① It is very accurate.
All the inputs are properly stated. So the models become accurate. If we can generate properly converged models, they become really accurate.

② Time alignment is done automatically and optimally.
It doesn't depend on whether you speak slow or fast. The stress levels also might be different. HMM will still be able to process it.
Since we are taking "states" into consideration and their transition matrix, if the diagonal has bigger values, it means we want to be in the same state. So time alignment is automatic, unlike simple neural network models.
The change of state happens with Viterbi algorithm. Viterbi algorithm aligns the most optimal state sequence. So time alignment is optimal also.

③ It is mathematically tractable and is flexible in modelling real life phenomena.
We should be able to adapt the model to different phenomena. HMM ~~gives~~ allows us to use mathematics and probability and gives freedom to change the parameters so that the variations of phenomena will be absorbed by the model.

④ Once properly trained, it is statistically stable and performs the recognition task at near ~~or~~ real time or real time over a large set of varied utterances.
Here we have N^2T operations per word. So, it is very fast, unlike Neural Network models.

(5) It can be used for speaker independent cases. We can collect 50 speakers' samples and test it on 51st speaker. The model will still work and give nice results.

Flow of assignment → We are going to build the model using 20 utterances for every digit and for every speaker i.e. no of utterances * no of digits * no of speakers. Then we are going to test on all the speakers on other people.

Disadvantages of HMM →

(1) The no of states have to be carefully selected. As, once taken, it can not be changed for that model.
When N=5, we expect that four significant changes are going to take place.

(2) Training of HMMs require huge amount of training data over a large number of speakers or utterances, typically 500 or more.

(3) Choice of initial or starting HMM is critical for proper convergence of the model. Usually a no of starting HMMs are used and the model that gives the best performance i.e. $P(O/\lambda)$ is taken.

First we have to map N and M into the phenomena.
If we try to recognize a word, N will be less.
But if we try it for a sentence, N will be more.

Lecture - 34

03/11/21

Dynamic time warping →

It tries to forcefully match the salient part of the data. It decides which frame of test data should be matched with which frame of training data.

Two concepts →

① Features –

It is the information in each signal which has to be represented in some manner.

② Distances –

Some form of metric has to be used in order to obtain a match path. There are two types of distances i.e.

1> Local distance - a computational distance between a feature of one signal with the feature of another signal.

2> Global distance - the overall computational distance between an ~~en tier~~ entire signal and another signal of possibly different length.

The question of optimal feature extraction is not a trivial one. It is an open question and an open research problem.

We shall use Tokhura's distance or Euclidean distance.

It is also required to declare a backtrace array to keep pointing to the preceeding point in the optimal path

e.g → SPEECH vs SSPEEhH

These two should be matched.

The constraints that we shall impose on the algorithm are as follows (reasonable).

(5)

① Matching paths cannot go backward in time.

② Every frame in the input must be used in a matching path.

③ Local distance scores are combined by adding to give a global score.

For now, we will say that every frame in the template and the input must be used in the matching path. This means that if we take a point (i,j) in the time-time matrix, then the previous point must have been $(i-1,j)$, $(i-1,j-1)$, $(i,j-1)$.

The key idea here is that we just continue with the lowest distance path from $(i-1,j)$, $(i-1,j-1)$ or $(i,j-1)$.

The DP technique guarantees to find the lowest distance path through the matrix while minimizing the amount of operations. It operates in a time-synchronous manner meaning that each column in this time-time matrix is considered in succession which is equivalent to processing the input frame by frame so that for a template of length N, the maximum number of paths being considered at any time is N.

If $D(i,j)$ is the global distance upto (i,j) and the local distance at (i,j) is given by $d(i,j)$ then $D(i,j)$ will be -

$$D(i,j) = \min\left[D(i-1,j-1), D(i-1,j), D(i,j-1)\right] + d(i,j)$$

Given $D(1,1) = d(1,1)$ (initial condition), we have the basis for an efficient recursive algorithm for computing $D(i,j)$. The final global distance $D(n,N)$ (n for reference data and N for test data) $D(N,n)$ is the result.

$D(N, n)$ is the optimal distance which takes a valid path from the starting point to ending point.