

2018130889 이윤민 영어음성학 정리 누적9회차 (12/11일 제출)

(1 회차~10 회차까지 순서대로 있고, 9 회차 정리는 11~12p 입니다.)

2018130889 이윤민 영어음성학 정리 1 회차

***Nasal(비강) tract / Oral tract velum 과 관련**

*Nasal tract open (코로 숨쉴 때): velum 은 lowered 되고 그에 따라 Nasal tract 가 열려 코로 공기가 지나가게 된다. 즉 Velum lowered Nasal tract opened, Velum raised Nasal tract closed. (모든 모음 & 비음제외 모든 자음)

***Voiced / Voiceless sounds 모든 소리는 유성음과 무성음으로 나뉜다.**

-유성음(성문 열림/공기저항 x 통과): 모든 모음 & 일부 유성자음 (g,d,b,m,n,z etc...)

-무성음(성문 닫힘/공기저항 o 통과): 유성음을 제외한 모든 소리. (k,t,p,s,f,h etc...)

***Oro-nasal process in velum (엑스레이 하얗게 나오는 부분이 뻐.)**

-Constrictor (협착을 만드는 주체)의 종류: Lips, tongue tip, tongue body

Constriction 을 만드냐와 그 degree(CD)(상하)에 따라 control 가능

Constriction location(CL)(앞뒤)로도 control 가능

Cf)모든 모음은 Constrictor 로 반드시 tongue body 만 사용. Tongue tip 이나 Lips 는 존재하지 X.

Ex)Tongue tip 쓴다 할지라도 조금/많이 막을건지 (degree), 앞/뒤로 갈건지 (Location) 에 따라서 변화가 생기는데, 이렇게 소리를 보다 자세하게 specify 할 수가 있다.

***모음의 종류 - 모두 Approximants(근접/접근음): 조음기관이 서로 근접은 해도 직접 닿지는 X.**

-Vowel(모음)은 무조건 Approximants. (막힘이 없는 게 모음의 정의이기 때문.)

-Consonant(자음)은 그렇다면 Approximants 인 것도, 아닌 것도 존재. (아래에서 다룬다.)

***자음의 종류 - 3 가지. 자음은 조음 방법에 따라 다음 세가지 중 하나에 속한다.**

1.Stops 폐쇄음(나오던 기류가 일단 완전히 막혔다가 터져나오는 소리) : p t k

2.Fricatives 마찰음(두 조음기관의 간격을 아주 좁히고 그 사이로 폐에서 나오는 공기를 스쳐나가게 발음하는 소리) : s z f v th dg

3.Approximants 접근음(조음기관이 서로 근접은 해도 직접 닿지는 않으면서 나오는 소리) : r l w j

***Phoneme (음소)**

-음소(어떤 언어에서 의미 구별 기능을 갖는 음성상의 최소 단위):

Ex) sip 에 쓰인 /s/와 zip 에 쓰인 /z/가 두 개의 다른 음소.

-영어의 /f/와 같이 우리의 머릿속에 저장되어 있는 것이 바로 이 phoneme.

2018130889 이윤민 영어음성학 정리 2 회차

*How to produce English consonants & Vowels.

Ex) n 을 발음하면서 코막으면 소리가 막힌다. (입에서는 이미 완전막힘이 있으므로 코까지 막으면 더 이상 바람이 새어나갈 수가 없기 때문.) 즉, n 과 같은 Nasal sound 는 velum lowered 돼서 코로 바람이 통하여 나는 소리.

*Complex tone in spectrum

*Spectrum 분석: "어떤 시점에서 어떤 주파수가 많구나~"를 분석하는 것. Complex 한 형태를 복잡하게 내버려두는 게 아니라 $1+1$, $1+5$ 와 같은 식으로 단순하게 쪼개어 표현하는 것. (푸리에 발견). Ex) 어떤 형태의 sin wave 를 세가지 형태의 sin wave 들이 합쳐진 것으로 분석 해 볼 수 있다. 예컨대 sin wave1 signwave2 signwave3 (Frequency:각 100 200 300 hz, Magnitude:1>3>2)인 세 사인곡선의 각 지점 지점마다의 숫자들을 모두 합하여 하나의 곡선(sinwave4)으로 나타낼 수 있다. 즉 여러 다른 sin wave(simplex tone)들의 합은 sin wave 가 아닌 복잡한 소리(complex tone)로 표현되며, 그런 식으로 소리 만들 수 있음. 복잡한 소리에서도 반복되는 주기가 보이며, 이 주기는 가장 low frequency 의 주기와 일치.

*하나의 sin wave 는 Frequency 와 Magnitude 로 표현할 수 있음. (X 축:Frequency, Y 축:Magnitude=Amplitude) (cf:sin wave 의 X 축:시간, Y 축:Value 값) 이렇게 sin wave 를 다른 domain 으로 표현하는 방법도 있음. 단순한 두-

소리는 한가닥의 spectrum 으로만 표현되고, 일반적으로 우리가 듣는 복잡한 소리들은 (X 축이 가득 채워져서)여러 가닥으로 표현됨. Sin wave analysis

***Spectrogram**: spectrum(시간개념 x)을 time 으로 visualize 한 것. (X 축:시간, Y 축:Frequency) 이 시점에 어떤 성분이 많은지, 주파수별로 어떤 성분이 많은지 파악할 수 있다.

***어떤 소리의 pitch** 는 그 소리의 가장 낮은 Frequency 의 tone 과 일치한다. (단위는 hz 사용) 가장 slow 한 tone 의 Frequency 가 우리 말의 음높이(pitch)와 동일하다는 뜻. 우리의 Vocal fold 와도 일치. 또한, 한 소리의 뾰족한 막대들은 완전한 등간격이다(Ex: 126hz-252hz-378hz...). Hertz 는 등간격이지만, Amplitude 은 gradually decreasing 하는 상태. 즉, Amplitude 는 점점 감소, Frequency 는 등간격으로 점점 높아짐 (=빨라짐). 이게 Source.

***Source 와 Tube**: 우리의 성대에서는 늘 똑같은 소리가 나며(source, larynx), 성대 위에 입모양을 가진 머리뚜껑에 의하여(Tube) Filtering 이 되어 결과적으로 다른 소리들이 탄생함.

***에너지**:저주파:높음고주파:낮아짐. 복잡한 소리에서는 전반적으로는 에너지가 내려가는 양상을 보이지만 중간중간에 에너지가 높아지고 낮아지는 현상을 볼 수 있음. (Harmonix 은 등간격 유지)

2018130889 영어음성학 정리 3 회차

*코딩:기본적으로 자동화라 생각.

왜 자동화?: 똑 같은 게 반복되니깐 기계화시키는 것. (매번 일일이 할 필요 X) ex:휴대폰과 컴퓨터태블릿 등의 프로그램.

*Programming language: 만국공통어가 되는 게 X. 여러가지 특성/장점/단점 존재.ex)C 언어,Python,R 언어. 그런 모든 language 도 다 다르지만 공통의 속성이 있음. 인간의 language 와도 공통점을 가지며 그 공통점이란 단어와 문법을 일컬음. 단어란 정보를 담는 그릇이며 그 단어 안에 사과라는 정보를 담으면 사과라는 단어가 되며, 그 그릇에는 여러가지가 바뀌며 담길 수 있다. 따라서 숫자와 문자를 담는 variable 이 필요함.

단어(정보를 담는 그릇)에 해당되는 것:변수

이 변수에 숫자/문자같은 정보 담을 수 있음 --

>기기한테 이야기할 때 communication 하는 방법이 바로 문법에 해당. 기계의 문법은 사람만큼 복잡하지 않음.

*기본적인 특징 4 가지. 때

1.변수(variable) 이라 부르는 그릇에 정보를 넣는 것. (빈 그릇에 정보 assign 하기)

어떤 language 를 사용하더라도 변수에 정보 assign 하는 것 변함없음. variable assignment.

2.자동화/기계화라고 생각할 때 직관적으로 떠오르는 것: "~할 때 이렇게 해 달라"라는 조건. 이 조건이 당연히 필요할 것. 이 Conditioning 에 대한 문법 필요.

모든 cpu 문법은 사람 말처럼 if 문법을 쓴다.

3.자동화의 가장 중요한 것 중 하나가 여러 번 반복하는 것. 이거는 for(몇번동안 반복하라)라는 문법을 씀. 이건 어떠한 language 도 다 쓰고 있음. For loop. 계속 반복시키는 것.

->이 세가지가 프로그래밍의 공통적인 문법

4.(가장 중요) 3 번까지는 개별적인 문법임. 결론적으로 말하자면 함수를 배워야 함. 함수:중간에 어떤 과정이 들어갔는 지는 모르겠지만,

어떤 걸 넣어서 어떤 값 나오는 거. 내부적으로 뭔가 엄청나게 복잡하게 돌아갔을 것. 입력에 해당하는 것)우리의 마우스클릭. 출력)

*함수란? 입력에 어떤 것이 들어가서 출력이 되는 것. (함수 내부에 들어가는 것은 다양함) ex)Praat, 자동차, 두 개의 자연수를 주면 시작부터 끝까지의 합을 구하는 함수 등. 함수 속에도 또다른 함수가 들어갈 수 있으며 이런 식으로 함수를 반복과 재사용을 할 수 있음. 코딩도 이런 방식으로 이루어진다.

*코딩 용어와 사용법

-Variable: 컴퓨터에 주문할 때 주는 정보. 이걸 사람의 말과 컴퓨터의 정보에 담는다. 숫자/문자.

-Equal sign(=): 같다는 뜻이 X. 오른쪽의 정보(숫자/문자)를 왼쪽의 variable 에 assign 한다는 뜻.

-Run: Run 을 해야 variable 에 정보(숫자/문자)가 assign 되는 행동이 완료됨.-

Print: 어떤 변수(ex: (a))를 넣으면 그 안에 있는 것을 print out 하는 함수. Print 가 어떤 함수인지는 미리 알아야하며 구글링을 통하여 알 수 있음.

2018130889 이윤민 영어음성학 복습 4 회차

*어떤 variable 의 내부 정보를 나타낼 때: 대괄호 [] 를 사용하며, 이 대괄호 내부에는 index 를 넣는다. dict 에서는 왼쪽 표제어를 index 로 사용하며, 표제어는 문자/숫자 둘 다 가능하다.

Ex) a = [1, 2, 3]일 때, print(a[0])은 a list 의 0 번째 element 를 들고 오라는 의미이며, 따라서 1 이 결과로 나오게 된다. Ex)A = '123'; print(type(a)); print(a[1]) <class 'str'>, 2. 여기서 2 는 숫자가 아니라 문자이다. 주의할 점은 컴퓨터에는 숫자가 1 이 아니라 0 부터 센다는 것이다. 영에서 시작하여 오른쪽으로 갈수록 숫자가 증가하며, 끝에서부터 세면 마이너스 1 에서 시작하여 왼쪽으로 갈수록 절대값이 커진다. 어떤 index 이던지간에 첫번째 index:0, 마지막 index:-1.

*Float 함수/ int 함수: 어떤 variable 이 들어오면 그것을 float type/int type 으로 바꾸어 줌. 예를 들어 a = 1 (int); a = float(a); print(type(a)) 의 결과값은 float. 원래 int 였던 1 이 함수를 거쳐 float 으로 바뀌었기 때문이다. 반대로 a=1.2; a = int(a); print(type(a))를 하면 int 가 나온다. 참고로 print(a)를 한다면 반올림되어 1 이 나오게 된다.

*List 함수: 강제로 list 로 쪼개어주는 함수. Ex) a = '123'; a = list(a); print(type(a)); print(a); print(a[2]) <class 'list'>, ['1', '2', '3'], 3. 주의)문자는 항상 ' ' 속에 들어있으며, 아무것도 감싸고 있지 않으면 숫자로써의 정보이다. Tuple 도 같은 방법으로 다룰 수 있음. 지금까지는 내부 data access 해봄..

*Str 과 list 는 내부 정보 access 하는 과정이 거의 같다. S = 'abcedf' (str), n = [100,200,300] (list).

*range 함수: 여러 개의 정보를 한꺼번에 가져올 수 있다. Ex) [1:3] 은 첫번째 것~세번째 것 직전의 것 (!)을 나타내는 표현이다. 즉, 첫번째 두번째 것을 가져온다. (0 번째와 첫번째는 다르다!) ex) [1:] 첫번째~끝까지. Ex)[:3] 0 번째~3 직전까지. (닫는 숫자는 늘 끝에서 숫자 하나 빼서 생각하기)

*lens 함수: variable 의 길이 (length)를 알려주는 함수.

Ex) s = '123456'; print(lens(s)) 6, ex) n = [100, 200, 300]; print(lens(n)) 3

*upper 함수: 소문자를 대문자로 바꾸어주는 함수. 함수의 형식이 다르다. S = 'abcdef'; s.upper() 'A BCDEF'. 와 같이 .을 먼저 찍고 함수이름을 적어준다. 이렇듯, 어떤 글자를 적고 (괄호)를 뒤에 붙이면 그것이 함수역할을 한다. .을 찍는 방법으로도 함수를 만들 수가 있으며, 마침표 앞에 variable 의 이름을 적으면 된다.

*find 함수: 무언가 찾아주는 함수. Ex) S.find('house'): 변수 안에 담겨있는 str 속에서 'house'를 찾아라 이 예문에서 house 는 11 번째에 있으며, 이 때 띄어쓰기도 함께 센다.

*strip 함수: 잡스러운 것들을 지워주고, 순수 text 만 남겨주는 함수.

*Split 함수: str 에 문장이 들어가있을 때 이를 단어별로 잘라주는 함수. 결과적으로는 단어의 list 로 만들어준다. S.split(' ') : (' ')안의 것을 기준으로 잘라주는데, 지금의 경우에는 space 이다.

(이후의 복습 과제는 jupyter notebook 으로 직접 필기하여 정리했습니다.)

2018130889 이윤민 영어음성학 정리 5 회차

*Numpy 함수: List 상태에서는 mathematical 한 계산/작동이 불가능하다. 그래서 그걸 하고싶을 때 Numpy 라는 함수를 씀. 즉, Numpy arrange/Array : list 를 operable 한 행렬/벡터로 만들어 주는 것. -

예컨대 a = [1,3,5] , b = [2,4,6]과 같은 두 개의 list 가 있을 때, numpy 를 적용하지 않았을 때와 적용했을 때 다음과 같은 차이가 있다. (이 import 과정을 거쳐야 numpy 함수를 사용할 수 있으며, numpy --> np 로 축약하여 적어도 된다.) 이때의 type 은 numpy.ndarray 이다.

*행렬: [[1,2,3], [4,5,6]]와 같이 큰 list 안에 작은 list 을 넣을 수 있으며, 이런 형식을 2 행 3 열 즉 (2,3)의 행렬이라고 하고 가로 세로 행/열 에 대한 직사각형 숫자 array 의 형태로 나타난다.

*import numpy 의 두 가지 방법: 1)Numpy.A.d.f (큰 껍데기-

>작은 껍데기 순서로 적자), 2)From numpy import A (Numpy 에 있는 A 를 불러오자)

*Dtype: dtype 을 통해 내부 정보의 형태를 고정할 수 있다. ex) dtype = 'int'라고 설정하면 i nt 값들이 나오게 된다.

*Zeros/Ones: 행/렬의 직사각형을 만들어 주는데 그 내부는 0/1 로 차 있다. ex) `Np.zeros([2,3])`
zeros 라는 함수에 (2,3)라는 list 를 만들고, 이 내부는 0 으로 채워짐.

*Float type (ex: float32, float64): 1.000000...와 같이 소수점이 32 비트/64 비트 차지할 만큼 내려간다. 숫자가 클수록 정밀한 정보를 표현할 수 있지만, 저장하는 데이터의 양이 많아진다는 반대급부가 있다. 즉, 정확도와 데이터를 쓰는 메모리 양은 반비례한다.

*`np.arange(5)` -> `array([0,1,2,3,4])` , `np.arange(0,10,2)` -
> `array([0,2,4,6,8])`: `np.arange` 함수를 통해 이렇게 index 들을 만들어낼 수 있다. 후자와 같은 형식은 0 과 10 을 2 의 증가분만큼 띄워서 나누라는 뜻이다.

*Linspace: linear space 의 줄임말. `np.linspace(0,10,6)` -
> 0 과 10 을 똑같이 6 개의 숫자로 나누어주라는 뜻이고, 여기서는 10 도 포함한다는 것을 기억해야 한다.

*벡터는 1 차원, Matrix 는 2 차원(직사각형), 직육면체는 3 차원이다. 이 직육면체의 3 차원도 행렬로 표현이 가능하다. 마지막 대괄호 개수와 차원은 일치한다.

*Normal: Normal distribution, 즉 정규분포의 데이터를 만들어주는 함수이다. `data = np.random.normal(0,1,100)` 일 때 평균이 0 이고 standard 는 1, 그리고 100 개의 random 한 데이터를 만들어내게 되며, 이를 bin 함수를 사용하여 히스토그램으로 표현할 수도 있다.

2018130889 이윤민 영어음성학 정리 6 회차

*`Np.array` 안에 list 를 넣어주면 numpy 형태로 바뀌고 이제 계산을 할 수 있는데, 이 때 다차원의 array 를 만들 수도 있다. (1 차원:벡터, 2 차원:정사각형 matrix, 3 차원:volume 등등)

*Sound 표현하기: sound 를 praat 로 분석할 때 pure tone 들의 합이 복잡한 sound 를 만들어낸다는 것을 배웠는데, 이 때 pure tone 들은 단순한 sine/cosine waves 이며 이러한 sinusoidal function 들을 만들어 내는 것을 phasor(파도 물결모양)이라고 한다. 즉 sine(태극문양), cosine 은 다 phasor 이다. ($\sin 0 = 0$, $\cos 0 = 1$)

*Radian: Sine, cosine 에 들어가는 입력은 각도(degree)가 아니라 반드시 radian 값으로 변형해 주어야 한다. 그렇다면 radian 이란? 두 직선의 교차점을 중심으로 하는 원을 그렸을 때, 원주가 두 직선에 의해 잘린 원호의 길이를 원의 반지름으로 나눈 값으로 차원이 없는 양이다.

*오일러(Euler) 공식: E 의 θ i 승 = $\cos(\theta) + i \sin(\theta)$ | \cos 과 \sin 이 다 들어가 있음

-

여기서는 θ 가 입력값인데, 그렇다면 e 와 i 는 무엇일까? e 는 자연 log 의 base 이자 2.71...

처럼 파이와 같이 계속 가는 무리수이며, 즉 정해진 상수값이다. i 는 imaginary 의 약자이며 즉 real 의 반대이다. 여기서 짐작할 수 있듯 실수의 반대, 즉 허수이다. 허수는 루트-1 이고 따라서 제곱하면 -1 이 나온다.

* $F(\theta) = e$ 의 세타 i 승. Radian 값이 변하면서 값이 달라지는 함수라고 생각하면 됨. f 라는 함수에 θ 가 들어가는데 그 함수의 생김새가 e 의 θi 승이다. 그렇다면 여기서 어떤 결과값이 나올지는 어떻게 알까? 어차피 다 숫자값들이니까 결과값 또한 숫자값이 나올거라는 추측이 가능하다. 따라서 세타값에 따라 다른 숫자값들이 나올 것인데, 이 때 나오는 다양한 결과값들을 포괄하기 위하여 우리는 복소수($a+bi$) 개념을 사용한다.

*복소수 plotting 하기: 복소평면(복소수 평면, complex plain)은 X 축과 Y 축으로 구성되어 있는데 이때 복소수 $a+bi$ 중 X 축이 a , Y 축이 b 에 해당한다. ($1 = a:1 \ b:0$ / $i = a:0 \ b:1$ / $-1 = a:-1 \ b:0$ / $-i = a:0 \ b:-1$)

1) 이걸 좌표에 다 찍어보면 원이 그려진다. 이 그래프를 어떻게 사용해볼까? ex) 아무 radian 값을 넣었다고 쳐 보자. 예를 들어 0.3, 0.8 이라고 치면 저 값이 주는 건 $0.3+0.8i$ 라는 것이다.

*모든 데이터는 벡터화되어야하는데, 이 때 벡터는 (1,0), (0,1)과 같은 숫자열이며 이렇게 벡터값으로 표현이 되는 열들은 세타값에 따라 반시계방향으로 바뀌게 된다.

*Projection: 이 개념은 말 그대로 flashlight 로 비출 때 어떤 모양이 나오느냐와 관련된 개념이다. (빛 비춰서 2 차원으로만 본다고 생각하자)

-

ex)x 축으로 빛 쏘면 왼쪽오른쪽으로 왔다갔다~ y 축으로 빛 쏘면 아래위로 왔다갔다~하는 형태일 것.

*Projection 의 두 방향에 따라 볼 수 있는 관점이 달라진다. (실수/허수)

-따라서 실수의 관점에서만 보겠다면 위에서 빛을 쏘고, 그 빛은 원-

으로 왔다갔다 할 것이며, 실수는 보기싫고 허수만 보겠다면 옆에서 빛을 쏘고, 그 빛은 아래위로만 왔다갔다 할 것이다.

-

허수만 본다면 sin, 실수만 본다면 cos 이다. 즉, 오일러 phaser 은 sin 과 cos 을 동시에 가지고 있고, 복소수와 오일러숫자의 특이한 성질을 이용해서 sin 과 cos 동시에 나타내어 어떨땐 a 로, 어떨땐 b 로 projection 해서 어느 한쪽만 관찰할수도 있음.

*Pure tone 에서 우리가 넣는 입력값은? 1.진폭, 2.Frequency(1 초에 몇번 움직이는가/왔다갔다하는가) 만약에 우리가 sine 세타라고 쓴다면 그 각도값이 변한다고 했을 때 시간의 개념이 들어갈까? 들어가지 않는다. 하지만 진정한 소리를 만들어 내려면 각도 개념과 더불어 그 속에 초 개념을 함께 넣어주어야 한다. 즉, 실체의 소리는 반드시 그 속에 시간 개념이 들어가야 함. (sr 과 dur 로 time 을 만들어낼 수 있으며, 이 t 는 어차피 계산을 해야하는 값이므로 np 를 사용해야 한다.)

2018130889 이윤민 영어음성학 정리 7회차

*Wave: sin wave 여러 개 쌓여서 \diamond 성대 소리를 표현할 수 있다. 이게 우리가 보는 wave. X 축은 time, Y 축은 value/energy

*Wave form: wave 를 frequency domain 으로 볼 수 있음.

*Spectrogram: 어떠한 구간에서 어떤 frequency 성분이 많은지를 보여줌. *Spectrum: Spectrogram 에서 한 given time 을 slice 로 잘랐을 때를 보여줌. 즉, spectrum 을 시간상으로 쭉 쌓으면 \rightarrow Spectrogram 이 된다.

*Formant 다루기

*기본 성대 소리를 Format 으로 잘라보기 \rightarrow Spectrogram 을 산처럼 빙글빙글 자르는 것 \rightarrow 이렇게 하면 아~이~소리 (모음 소리 만들어냄)

*각 산맥의 특성에 따라 다른 모음 소리가 만들어진다. 그 중, 특히 첫번째/두번째 format 이 중요하므로 F1, F2 를 설정하는 작업이 중요하다.

-Gradually decreasing 하는 Spectrum 에서 산맥을 하나하나 만들어야 하는데, 첫 산맥을 만들고 (주파수도 함께 설정) 그 오른쪽에도 연속적으로 산맥을 계속 만들어 주어야 한다. 이 작업을 통해 성대 소리에서 사람의 발음처럼 소리가 바뀐다.

*anaconda prompt 로 소리를 만들어보자 (tip:순간적으로 나오는 *기호 : 내부적으로 실행된다는 뜻)

-time 만듦 \rightarrow Time 으로부터 각도(phase)만듦 \rightarrow 이것을 sin 함수에 통하게 만듦

-Amp 설정: $s = \text{amp} * \text{np.sin}(\text{theta}) \rightarrow 440$ -s playout 하는 수식: `lpd.Audio(s, rate=sr) *hz` 와 관련한 소리의 변화: ex)440-880-1760u... 옥타브만 높아질 뿐, 전부 같은 '라'소리. 즉, 같은 배수의 소리는 같은 음을 가진다.

*Sin 대신에 cos 을 쓰면 시작점이 $0 \rightarrow 1$ 로 변경된다. 그러나 1 초동안 왔다갔다 하는 횟수는 같기 때문에 sin/cos 의 차이로 소리가 달라지지는 않음. (똑같은 '라' 소리) 즉, shape 자체는 같지만 $1/2\pi(90^\circ)$ 정도의 이동한 모양 차이는 존재한다. 따라서 이러한 이동으로 인해 소리는 달라지지 않는다는 점을 유추할 수 있다. 이러한 각도를 phase 라고 하며, 이 phase 는 우리가 듣기에 전혀 차이를 낳지 않기 때문에 어떠한 sensitivity 도 갖고 있지 않으며 따라서 이는 우리가 인식

하지 못한다. (차이를 느끼려면 Frequency 조정)

*Complex phasor 실행 C 라고 명명(complex 자체는 plotting 이 안되므로 a b 각각을 사용해서 2 차원으로 나타내어 plotting 을 함) *Resonance 의 첫번째 입력으로 sampling rate 사용. *Width: 산맥의 shape (뽕죽/똥똥) 결정.

2018130889 이윤민 영어음성학 정리 8회차

*행렬 계산법: 3×2 matrix(행렬)이 있다고 생각하자. 이 행렬 오른쪽에 곱해질 수 있는 건 $2 \times ?$ 행렬 뿐이다. 오른쪽 column(세로열)들은 왼쪽의 row(가로열)개수와 똑같아야 한다. 즉, 행렬끼리 곱할 때 인접한 곳의 숫자는 똑같아야 한다. (오른쪽 column들을 왼쪽 세로열에 곱하고 각각 더해주는 게 행렬이 곱해지는 원리이다.) Ex) 3×2 와 2×3 의 행렬을 곱하면 결과는 $3 \times 3 \rightarrow$ 바깥쪽에 나온 숫자들끼리의 행렬이 된다. 앞의 원리에 따라서 만약 3×2 와 2×1 의 행렬을 곱하면 결과는 3×1 이 될 것이다.

* $A \cdot x = b$ 에서 A에 해당하는 부분이 인공지능(기계함수), x가 입력, 그리고 b가 출력이다. 만약 여기서 벡터(x)를 A앞에 두고 싶으면? Transpose해 주면 된다. 예를 들어 2×1 과 3×2 는 곱해지지 않는다. 따라서 각 행렬을 1×2 와 2×3 으로 바꾸어주면 된다. Dimension만 같으면 transpose한다고 행렬의 결과값이 달라지지는 않고, 이렇게 곱해진다는 건 함수가 적용되었다는 뜻이다.

*Vector(벡터)란 어떤 차원에서의 한 점이며, 행렬에서 숫자가 아무리 많아져도 차원만 늘어날 뿐, 좌표상에서 한 점으로 찍힌다는 특징은 동일하다.

*Scaler: 행렬 앞에 곱해지는 어떠한 상수값을 scaler이라고 하며, Scaler도 일종의 matrix(1×1)이다.

*행렬 두개가 더해지면 기하적으로 좌표에서 두 점과 원점에서 끌어당기는 힘을 상상할 수 있다.

*Column space: 예를 들어, 3×2 에서 3이 column이 되며, 이 column차원 벡터가 represent되는 곳은 3차원이다. 이 column점들과 원점을 찍고 색칠하면 plain위의 삼각형이 생기고, 이 삼각형을 각 변을 기준으로 넓게 펴서 밀어보면 전체 space가 생긴다. 이 확장하는 과정을 Spanning이라 이름 붙이고, spanning의 결과로 2차원의 plain이 생긴다. 따라서 column space는 2차원이다. (lineal combination으로 만들어진 점은 나머지 점들이 spanning해서 만들어진 column space 위다.)

*Whole space: 위의 예시에서 whole space는 3차원이다.

*위 예시에서 유추할 수 있듯이 Column space(2차원)는 항상 whole space(3차원)보다 같거나 작다.

*Null space: Whole space(3차원) - Column space(2차원) = Null space(1차원). Null space를 기하적으로 계산 해 보자. Spanning된 2차원이 plain으로 있다고 했을 때, 이 plain과 수직 관계에 있는 한 선(원점 통과)을 상상할 수 있고, 이 선은 1차원이기에 1차원의 null space가 결과로 나온다. 즉, column space가 whole space를 채우지 못하면 orthographical한 게 생기고, 그것이 null space.

*Row space: Column space가 있다면 row space도 있을 것. 이걸 Row vector를 기준으로 만든 space다. Column space를 만들어내는 과정과 동일하며, 마찬가지로 whole space이하의 차원이다.

*Independent/dependent vectors: 예를 들어, a b c 세 개의 벡터가 있을 때 $?*a+?*b=c$ 가 나온다면 c는 dependent vector, 즉 같은 선상에 위치한 vector. Independent vector은 같은 선상에 있지 않는 vector이다. Dependent vector이 있다면 whole space의 차원이 줄어들게 된다.

2018130889 이윤민 영어음성학 정리 9 회차

*Null space: (선) vector: $Ax=0$ 행렬 을 만족시키는 모든 x 값들의 모임. (인공지능에서 중요 원리)

-Null space 는 보통 직선으로 표현되지만, 그렇다고 해서 Null space 가 반드시 벡터들의 연결에 수직인 선 위에 있어야 하는 것은 아니다. . (Null space 선과 평행한 선들의 vector 도 영향을 미치지 않는다.)

Null space 는 입력이 변하더라도 출력에 영향을 미치지 않는 값들이고, 바로 이 출력에 영향을 미치나/안미치느냐가 null space 여부를 결정하게 된다. 위 식에서 A 부분은 data 를 통해 학습이 가능한 인공지능 기계이다. (Null space 선과 평행한 선들의 vector 도 영향을 미치지 않는다.)

-Null space 은 왜 필요한가?: 인간은 진화하면서 null space 를 더 확보해왔다. 어떤 행렬에 대해 $Ax=b$ 라고 할 때, 출력(b)에 해당하는 부분이 인간이 하고자 하는(사냥,악기연주 등)일이고 그럴 때 task 을 이루는 데는 전혀 지장이 없지만 발생하는 위기 상황이나 자연 재해에 해당하는 것이 null space 이다. 즉, b 라는 task 를 수행할 때 피해갈 수 있는 공간이며, "입력이 많이 변해도" 출력값에 지장을 주지 않는 부분이 null space 이다. 물론 출력값에는 영향을 주지 않지만 의미있는 공간이기에 인간은 이 null space 를 늘리는 방향으로 살아야 한다.

*Vector: 물리학적인 관점의 "방향"이다. (하나의 값이라기보다는 방향에 대한 용어)

-Cosine similarity (cosine theta): 두 vector 가 얼마나 비슷한지를 수치적으로 알려주는 지표.

-ex) a vector, b vector 의 각도를 각각 구하고 그 각도의 cosine 값을 구하면 된다.

<Eigen Analysis> : 선형대수에서 필수로 배우는 개념.

*Eigenvector: 고유(unique)한 vector 이라는 뜻. 어떤 '변환'에 대해서 방향이 변하지 않는 벡터(방향)를 의미하며, 이는 하나의 값만 존재하는 것이 아니라 직선으로 표현된다. (인터넷 사이트에서 보라색 선 전체)

-ex) a_1, a_2 두개의 column vector 을 또 다른 두개의 eigen vector 로 바꾼게 S_1, S_2

-eigenvector space: 더 정확하게 표현한 용어. 모든 eigenvector(점)들의 집합(선)을 의미한다.

-Ex) $2*2$ 행렬에서는 2 개의 eigenvector, $3*3$ 에서는 3 개의 eigenvector 가 생긴다.

-이유: 반대쪽으로 (원점)쪽으로 줄이면 나오는 다른 하나의 방향(S_2)도 있기 때문에 여러 개가 생기게 된다.

*Eigenvalue: 각각의 eigen vector(s_1, s_2)에 대해 각각의 eigen value 도 2 개 따로 떨어져 나옴.

*Inner product : 벡터들끼리 계산(곱하기)될 수 있도록 matrix 을 transpose 하는 데에는 두가지 방법이 존재하는데, 그 중 계산 결과값이 하나의 scaler 로 나오게 변형시키는 transpose 방법이다.

2018130889 영어음성학 정리 10 회차 (가장 최근)

* $S_r=10000$ 일 때 Spectrogram에서 7000에 해당하는건 사실 3000과 똑같다. (왜냐면 첨부터 5000 까지 설정했기 때문이고, 그렇기에 5000까지만 정확하게 봐야 한다.)

* S_r 의 1/2까지만 Freq로 표현됨.

*성대 소리에서 아/이 와 같이 모음 소리로 바꾸는 과정:

1. 여러 개의 harmonix 갈수록 점점 줄게 하는 작업
2. 각각의 산맥을 만들어준다. (산맥의 높이는 점점 줄어야 함.)

*Spectrogram: spectrum(세로 막대가 쪽 있는 형태)가 세워져서 한장 한장 시간 간격으로 쌓이는 것이며, 이는 plotting 하면 색깔로 구분되어 3차원 그림으로 나와야 한다.

*Spectrum → Spectrogram 으로 만들기: 하나의 긴 Wave가 있을 때 한 given time에서의 freq성분을 분석하면 하면 한장짜리 스펙트럼이 나올텐데, 어떻게 여러장짜리 스펙트로그램을 만들어낼까?

1. 앞부분에서 조금만 잘라서 스펙트럼 만들
2. 조금 뒷부분에서 잘라서 스펙트럼 만들
3. 이렇게 한장한장씩 만들.
4. 0.001초 자르고 또 analysis하고~ 그게 합쳐지면 스펙트로그램.

*제곱 후 로그를 취하는 이유:

-제곱(Power): 전체 제곱을 하면 1보다 큰값들은 훨씬 커지고 1보다 작아지는 값들은 훨씬 작아짐. (그래도 절대 음수는 나올 수 없다.) → 값들이 선명하게 보임.

-로그(Log): 제곱 취한 것들에 다시 로그를 취하는 이유는 너무 크고 작은 수들을 우리가 생각할 수 있는 범위 내의 숫자들로 바꾸어 주기 위해서이다. (급하게 고주파 저주파 오르내리락하는 것을 적당히 만들어서 눈에 잘 보이게, 다루기 쉽게 할 수 있다.)

*함수의 형식:

-input:: input하려는 부분에 괄호

-output:: output하려는 부분에는 return이라고 적어야 한다.