Phonology는사람이 인식하는 차원이고 Phonetics는 사람이 인식하지 못하는 하위의 차원이다. 발화 시의 물리적인 현상을 그 자체로 바라보는 것이 Phonetics이다.

Vocal tract에는 Nasal과 Oral tract가 있다. Nasal tract로 가면 nasal sound가 되고 oral tract로 가면 oral sound가 된다. Larynx는 후두이고, larynx 사이의 틈을 glottis라고 한다. 사람이 꿀꺽할 때 기도를 막는 역할을 하는게 epiglottis이다. Nasal/Oral은 Velum에 의해 결정되는데, Velum이 raised되면 oral이고 lowered되면 nasal이다. 코로 숨 쉴 때 velum은 lowered된다. Voicing을 결정하는 것은 larynx에서 이루어진다. Glottis가 열려 있으면 성대가 떨리지 않아 voiceless sound가 나오는 반면에 glottis가 닫혀 있으면 성대가 떨려 voiced sound가 나온다.

영어에서 소리는 5가지 요소로 인해 결정이 되는데, Constrictor, Constriction Location(CL), Constriction Degree(CD), Velum, Larynx가 그것이다. Lips, tongue tip, tongue body는 articulatory process를 담당하는 constrictor이다. 이곳에서 혀와 입술을 이용해 소리에 영향을 미친다. Velum은 oro-nasal process를 담당한다. 이곳에서는 소리의 oral/nasal을 결정한다. Larynx에서는 phonation process가 발생하는데, voicing이 이곳에서 결정된다.

Constrictor이 정해졌다면 Constriction location, Constriction degree로 더욱 specify 해야 한다. Lips가 constrictor이라면 bilabial, labiodental이라는 CL로 나뉜다. Tongue tip이 constrictor이라면 dental, alveolar, palato-alveolar, retroflex로 CL이 나뉜다. Tongue body가 constrictor이라면 palatal, velar로 CL이 나뉜다. 'J(y)'는 palatal에 속하고 'sh'는 palate-alveolar에 속하고 'r'은 retroflex에 속한다.

Constriction Degree는 stops, fricatives, approximants, vowels로 나뉜다. Approximants에는 'r', 'l', 'w', 'j' 4개만 있다. 'm', 'n', 'ng'도 nasal sound이지만 stop이다. 모든 모음의 constrictor은 tongue body이다. 모든 phoneme은 이러한 요소들의 combination으로 설명이 된다.

Spectrogram에서 띠로 나타나는 것을 formant라고 한다. 각 띠마다 아래에서부터 first formant, second formant 등으로 부르며 f1, f2 등으로 줄여서 부른다. 아래에서부터 위로 formant는 무한대로 나타난다. Formant값이 모음을 결정한다. 같은 모음이면 비슷한 formant 형태로 나타난다. 즉 formant는 모음을 구별하는 수치적인 지표이다.

Pitch setting- pitch range에서 설정을 해줘야 남자 목소리, 여자 목소리 구별이 가능해진다.

소리를 녹음하고 확대하면 하나의 큰 파도, 패턴이 보이는데, 이 큰 파도 하나에 성대의 떨림 한 번이 대응된다. 큰 파도가 1초 동안 나타나는 횟수가 바로 pitch(Hz)이다. 이 큰 파도의 duration을 재고 이 값을 통해 (1 나누기 duration값)을 하면 큰 파도가 1초 동안 나타나는 횟수를 계산할 수 있으며 이것이 pitch값이다. Pure tone으로 내 목소리의 pitch를 입력하면, 내 목소리와 같은 높이의 소리를 들을 수 있다. 이때, pure tone의 곡선은 사인 곡선과 같다.

이 세상 모든 소리는 사인 웨이브의 조합으로 표현된다. 사인 웨이브는 frequency와 amplitude로 결정된다. 사인 웨이브의 x축은 시간이며 y축은 단순한 값(value)이다. 우리 주위의 모든 소리는 pure tone이 아닌 복잡한 소리라고 할 수 있다. 이 때 이 복잡한 소리의 spectral analysis에서 나오는 첫번째 사인 웨이브를 f0, fundamental frequency, the number of vocal cords vibration of a second 등으로 부른다. F0의 frequency가 전체 소리의 pitch이다.

Source는 성대에서 나는 소리를 그대로 녹음한 소리이다. 성대에서 소리를 바로 녹음하는 것을 EGG라고 한다. Source의 소리와 기타의 소리가 유사하다. Source의 spectrum은 pulse train의 형태를 띈다. 여기서 Pulse는 waveform에서 툭 튀어나오는 부분들을 가리킨다. Pulse train의 spectral analysis를 해보면 gradually decreasing하는 형태를 띈다. Source는 harmonics를 이룬다. Harmonics-> F0의 배음들(f0의 frequency의 배수들)이 이루는 사인 웨이브로 소리가 표현된다. 모든 소리가 harmonics를 이루는 것은 아니다. Source와 기타소리 등이 harmonics를 이루며 gradually decreasing을 한다.

Source는 vocal tract를 거치며 filtered된다. Source가 vocal tract를 지나 articulate되면 peak/mountain과 valley가 있는 산맥 형태의 spectrum이 만들어진다. 즉, 산맥을 만들어 주는 것이 filter의 역할이다. 각 모음을 발음할 때마다의 입, 혀 모양이 다른데, 각 모음마다 좋아하는 산맥형태가 있다고 생각하면 된다. Source가 vocal tract를 지날 때, 그 입모양이 도장 역할을 해서 source의 spectrum을 도장 찍는다고 생각하면 된다. 그렇게 해서 도장 찍힌 spectrum이 실제 발화된 소리의 spectrum이 된다. 이 때 이 spectrum은 산맥을 형성하게 된다. 그 때, 첫 mountain 부분이 첫 번째 formant로 f1이며 다음 formant가 f2이고, f1과 f2만 있으면 모음이 결정된다. F1을 y축, f2를 x축으로 했을 때 입 모양과 똑같이 된다. 이 때 f1은 혀의 높낮이를 결정하고, f2는 혀의 front, back를 결정한다. F2가 클수록 front vowel, f1이 작을수록 high vowel.

Spectrogram은 spectrum을 시간축으로 늘어놓은 것이다. Spectrum의 입체형이 spectrogram이라고 생각하면 된다. Spectrogram의 x축은 시간이고 y축은 frequency이고 짙은 정도는 amplitude이다. Spectrum들을 연달아 연속적으로 연결한 것이 spectrogram이다.

Spectral slice를 통해 spectral analysis를 정확히 보기 위해서는 최소 여러 개의 큰 파도(큰 파도 한 번에 성대 한 번 진동)를 선택해야 spectral analysis를 제대로 알 수 있다.

Stereo는 병렬적으로 일일이 다 더해서 나오는 소리이다. Pure tone이 남아있는 소리이다. Mono는 pure tone이 남지 않고 완전히 합쳐지는 소리이다. 그래서 단 하나의 wave form으로 표현되며, spectral analysis를 해보면 mono 소리를 구성하는 각 사인웨이브를 알 수 있다. Mono로 convert를 하면 mono 소리가 그 f0 소리와 pitch가 동일하게 된다. Stereo는 그렇지 않다. Gradually decreasing 하고 harmonics를 이루는 pure tone을 여럿 mono로 만들면 source 소리가 나와 pulse가 만들어지는데, pure tone이 많을수록 pulse가 더욱 두드러지게 튀어나온다.

코딩: 자동화하는 것
똑같은 것이 반복이 되기 때문에 자동화를 한다.
Ex) 폰에 있는 모든 것들은 코딩으로 이루어짐.

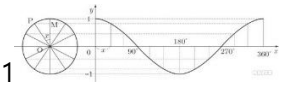
사람 language처럼 컴퓨터 language가 있다.
모든 language의 공통점-> 단어, 문법. 단어가 있고, 단어를 어떻게 combine 하느냐 하는 것. 단어는 그 속에 의미(정보) 포함. 단어: 정보를 담는 그릇.
컴퓨터 language에서 단어에 해당하는 것이 변수(variable). 변수에 정보를 담고 기계에 전달을 할 때에 그 방식이 필요함. 이때 필요한 것이 문법. 이때 문법은 다음과 같다. 첫번째, 변수에 정보를 넣는 것 variable assignment. 두번째, if문법을 씀. ~할 때는 ~하라 이런 것 if conditioning. 세번째, 여러 번 반복하는 것. For문법 사용해 여러 번 반복하게 함 for loop. 네번째, 함수를 배우는 것. 내가 무엇을 입력하면 어떤 출력이 나오는 것. 입력과 출력의 packaging을 하는 것이 함수.
정보의 종류에는 숫자, 글자. 두 가지만 존재.
'=' sign 뜻: 오른쪽에 있는 정보를 왼쪽에 있는 variable로 assign한다.
a = 1 하면 1이라는 정보를 a라는 variable에 넣는 것임.
print라는 것도 함수임. 어떤 변수를 그 안에 넣으면 변수에 assign된 것이 뭔지를 알려주는 함수임.
여기서 a가 입력이고 1이 출력.
a = 1 한 후 a = 2라고 또 하면 overwrite됨. 덮어씌워짐.
그냥 알파벳을 쓰면 무조건 변수로 취급함. 그러니 문자를 쓰고 싶으면 ''를 써야함.
Run 단축키는 시프트+엔터.
하나의 셀에서 마지막 줄에 변수를 하나만 쓰면 print 안 써도 그 변수에 assign된 것 보여줌.
여러 개를 한 변수에 담을려면 list 써야함. 이를 대괄호로 표현할 수 있음. a = [1, 2, 3] 이렇게 함. a = [1, 2, 3, 5, 'love']라고 해도 됨. List 안에 list가 들어갈 수 있음. a = [1, 'love', [1, 'bye']] 이래도 됨.
이때 대괄호 말고 소괄호 쓰면 type이 tuple이라고 나온다. Tuple이 보안에 더 강함. 사실상 list와 tuple은 같다고 보면 됨.
이것이 어떤 종류의 변수인지를 알기 위한 함수는 type임. type로는 list, int(integer), float, str(string) 등이 있음. Float는 실수(?)임. String은 문자배열.
a = {'a': 'apple', 'b': 'banana'} 이건 dict(dictionary)임. 이 때 'a'가 표제어, 'apple'이 설명어임. 'b'가 표제어, 'banana'는 설명어. Dictionary의 괄호는 중괄호.

a라는 리스트에서 부분만을 가져오고 싶다면 a[0]의 방식으로 가져올 수 있음. . a list에서 0번째, b에서 0번째 꺼를 가져와서 더한 것임. list에서 첫번째는 0번째임. 정보를 통째로 가져오려면 a를 적고, 그 중에서 선별해서 가져오려면 a[몇] 쓰면됨.
float라는 함수는 어떤 variable이 들어오면 그걸 float로 바꿔줌. int함수는 variable을 int로 바꿔줌. 정보에서 어떤 것을 가져올 때는 반드시 대괄호를 적고 그 안에 index를 적으면 내부적인 정보를 부분적으로 가져온다.
''로 되어있으면 문자 취급, 그게 아니라 그냥 숫자만 되어있으면 숫자 취급. '1'이면 문자, 1이면 숫자. Dict에 있는 정보를 access 할 때는 페어에서 앞부분을 index로 쓴다. 0,1,2 그런 숫자로 하지 않는다. String과 list는 정보를 접근하는 방식이 매우 비슷함. Parallel함.
s[1:3]이렇게 나온 것은 range로 정보를 가져오는 것인데, 첫번째부터 3번째 직전까지 정보를 가져오는 것임(3번쨰는 가져오지 않음.)
len함수는 정보의 개수, 길이를 알려줌(length).
더하기는 숫자만이 아니라 문자에도 똑같이 적용됨.
Upper은 업 시켜주는 함수임. 소문자는 대문자로 만들어줌.
find함수는 첫번째 나오는 것을 찾아주는 함수. 전부다 찾아주지는 않음. 띄어쓰기도 순서에 고려됨.
rindex함수는 가장 나중에 나온 것을 찾아주는 함수.
strip함수는 남은 것들 걸러서 온전한 텍스트만 남기는 함수.
s.split(' ') 뜻: s라는 string을 ' '(빈칸)을 이용해서 잘라라. 중요!
여러 개로 쪼개진 것을 다시 합치고 싶을 때 join을 씀
' '.join(tokens) 뜻: 빈칸을 이용해서 token에 들어 있는 list를 붙여라
s.replace('this', 'that') 뜻: 이 스트링에 있는 모든 'this'를 'that'으로 바꿔라.
For loop- 여러 번 반복 될 때 쓰임.
for i in a: in 뒤에 있는 것(a)을 하나하나씩 돌려서 i가 그 하나하나씩을 받아서 뭔가를 하라. a에 있는 것들을 하나하씩 돌려서 i에 할당을 하고 그 아래에 있는 것을 하라.
Range 함수: range 뒤에 어떤 숫자가 나오면 list를 만들어준다. 4라고 나오면 0부터 3까지의 list를 만든다.
Enumerate 함수: 번호를 매긴다. 그 list 값도 나오지만 추가로 번호를 매겨진다.
"{}: {}%".format(s, b[i]*100) 뜻: format괄호 안에 있는 것들이 왼쪽의 형태로 박혀라. 즉, 왼쪽의 형태로 format으로 하고 싶으면 쓰는 것.
Zip은 따로 독립된 함수를 합치는 것. 페어로 합침.
==하면 진짜 같다는 의미. =가 어떤 값을 변수에 assign하는 기능 이라면 ==는 진짜 같다는 equal sign임.
>= 이건 부등호. 순서 중요. a >= 0 이면 a가 0보다 크거나 같다는 뜻.
!= 하면 같지 않다는 의미(==의 반대)
For, if 함수에서는 끝에 클론(:)을 붙이고, 인덴트를 하는 것이 매우 중요.

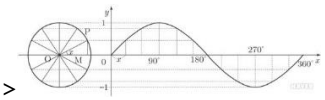
데이터를 숫자로 변환할 때 숫자의 열로 표현한다. 이 숫자의 열이 벡터. 사진들도 모두 숫자로 표현. 모든 데이터는 벡터의 형태로 되어야 한다. 데이터는 벡터의 형태로 했을 때 다루기가 쉽다.
행렬- 직사각형의 형태에 숫자를 넣어 배열한 것. 가로가 행, 세로가 열이다. 행렬은 벡터는 아님. 길게 늘여놓아 진 것이 벡터.
컬러 사진은 행렬을 세겹 겹쳐 놓은 것(RGB). 흑백 사진은 행렬 하나. 동영상은 시간에 따라 계속 행렬이 변하는 것. 흑백 사진은 2차원, 컬러는 3차원, 동영상은 4차원
소리도 벡터화 시킬 수 있다. 소리의 웨이브는 결국 점으로 이루어져 있는 것인데, 이를 숫자로 처리하는 것이다. 텍스트도 벡터화 시킬 수 있다.
중요한 함수는 import를 해서 그 라이브러리를 불러와야함. 여기서 라이브러리는 일종의 함수 패키지라고 할 수 있다. Numpy라는 패키지 안에 여러 함수가 포함되어 있고, Numpy안에 또 다른 함수패키지도 포함되어 있다. Numpy.A.D.f 이렇게 점(.)을 통해 아래 패키지로 접근해 나가는 것. 또는 from Numpy import A.D 를 하면 Numpy안에 있는 A, 그 안의 D를 불러올 수 있게 됨.
Numpy는 일반적인 list에서는 수학적인 처리가 안되기 때문에 수학적인 처리를 해줘야 할 때 쓰인다. Numpy가 적용되면 list끼리의 계산이 가능해진다.
Numpy를 이용해 벡터(1차원) 행렬(2차원) 등을 만들어 계산을 할 수 있다. Matplotlib.pyplot라는 라이브러리는 plotting을 할 때 쓰인다. 히스토그램을 만들 때 이 plt가 적용된다.
Numpy에서 중요한 것은 shape, ndim, dtype 등이다. Shape와 ndim은 깊이 연관되어 있다.
Csv 파일은 데이터를 주고받을 때 쓰이는 파일 형식이다. 컴마를 이용해서 구분하여 두개씩 짝지어진 데이터를 담는 파일 형식이다.
Numpy에서 데이터 타입은 float64 등과 같이 뒤에 숫자가 표현되는데, 여기서 숫자는 소수점 몇째 자리까지 표현할 수 있는지, 그래서 얼마나의 precision을 둘 수 있는지의 값이다. 이 값이 크면 더욱 정확하고 세밀한 값을 이용할 수 있지만, 용량이 커진다는 문제점이 있다.
Reshape을 할 때 중요한 것은 같은 size로만 reshape을 할 수 있다는 것이다. 예를 들어 (2, 3, 4)의 shape을 가진 array가 있을 때, 이것을 (1, 7, 3)등과 같이 size가 다른 형태로는 reshape할 수 없고, (4, 3, 2)와 같이 같은 shape일 때에만 reshape 할 수 있다. 또한 주목할 점은, (-1, 3, 2)라는 형태를 써서 reshape을 할 수 있는데, 이는 3과 2가 정해지면 남는 값은 4로 자동으로 정해지기 때문에, 남는 값은 -1로만 써도 제대로 적용이 되어 reshape된다.
A라는 variable이 numpy의 형태이라면, a.sum()을 하면 자동적으로 모든 값들이 더해진다. Np.sum(a)를 써도 같은 값이 나온다. 이때, a.sum(axis=0) 등과 같이 차원을 적용해서 sum을 해야할 때는 조금 복잡해지는데, axis가 0이면 첫번째 차원의 관점에서, 1이면 두번째 차원의 관점에서 sum을 각각 해야한다. 이는 잘못하면 헷갈릴 수 있으니 충분한 연습이 필요하다.
행렬 사이에서 계산을 할 때에는 두 행렬이 반드시 같은 shape일 필요는 없다. 같은 shape이 아니라도 broadcasting을 통해 계산을 할 수 있다.

사인이나 코사인 곡선처럼 생긴 것을 sinusoidal이라고 하고, 이런 sinusoidal function을 만들어내는 것을 phasor이라고 한다. 즉 sine function이나 cosine function도 phasor이 된다.
Degree: 0도에서 360도와 radian: 0에서 2파이와 일치.
사인 코사인 함수 안에 들어갈 값은 radian임. Degree 아님.
720도나 360도나 똑같음. 따라서 2파이나 4파이도 똑같음.

코사인에서 0이들어가면 1이 나옴. $\frac{\pi}{2}$는 0, π는 -1, $\frac{3}{2}\pi$는 0, 2π는 1



사인은 0이 들어가면 0, $\frac{\pi}{2}$는 1, π는 0, $\frac{3}{2}\pi$는 -1, 2π는 0 ->



이런 사인 코사인 함수들을 phasor라고 함. 코사인 함수도 phasor, 사인 함수도 phasor.
0부터 100파이까지 사인, 코사인은 몇 번 반복이 되느냐? 50번 반복.
오일러 공식: $e^{\theta i}$=cos(θ) + sin(θ) i 여기서 e는 상수값임. 2.71.... i도 상수값. 즉 θ(radian)값만 변함으로써 어떤 값이 나오는 함수임. $f(\theta) = e^{\theta i}$ 로 표현가능. 이 함수는 phasor임.
모든 수는 복소수 형태 a + bi로 표현 가능. 즉, $e^{\theta i}$도 a + bi로 표현 가능하다. 이때, 위 함수에서 세타 값에 0, $\frac{\pi}{2}$, π, $\frac{3}{2}\pi$, 2π 들어가면 각각 1, i, -1, -i, 1로 계속 반복되는 phasor형태가 나온다. 이 값들을 a가 x축, b가 y축으로 하는 복소 평면(complex plane)에 놓는다고 했을 때 (1,0), (0,1), (-1,0), (0,-1)로 반복 된다. 이것은 일종의 원 형태로 반복됨. 이때 그 점의 좌표와 x축이 이루는 각도가 세타(각도는 제1사분면부터 출발). 즉, 세타 각도 값만 알고 그 각도에 해당하는 부분을 원에서 찍으면 그 좌표가 바로 a + bi가 되는 것임. 이런 좌표들은 다 벡터이다.
Projection 시킬때. 위에서 보면 좌우로 왔다갔다하는 실수부분만 볼 수 있음. 옆에서 보면 위아래로 왔다갔다하는 허수부분만 볼 수 있음. 여기서 실수부분은 코사인만 보는 것이고 허수부분은 사인만 보는 것. 이것이 오일러 phasor.
이 오일러 phasor은 사인과 코사인의 성질의 동시에 가지고 있음. 우리가 원하는바에 따라서 어떻게 projection 하느냐에 따라 코사인이든지 사인이든지 원하는대로 볼 수 있다. 즉 훨씬 advanced된 phasor.
위 phasor에서 시간의 개념은 들어가 있지 않다. 그저 각도값의 변화에 따라 나오는 값들을 보여주고 있을 뿐이지 시간의 개념은 없다. 그러나 소리는 그 실체를 구현하기 위해서는 반드시 시간 개념이 있어야한다. 1초당 몇번 왔다갔다한다 그런 개념이 있어야 frequency가 정해지고 소리의 높이가 결정되는 것.

Sampling rate과 frequency간의 관계 -> sampling rate으로 표현할 수 있는 최대의 frequency가 있다. 이것이 Nyquist frequency다. 이 Nyquist frequency는 sampling rate의 절반이다. Nyquist frequency를 넘어가면 그 소리를 표현할 수 없다.

Amplitude의 개념을 사용하여 사인 함수를 만들려면, amplitude를 마지막에 곱하기만 하면 된다. Amplitude를 적용하면, 그 값에 따라서 진폭이 달라진다. 이때 이것을 오일러 phasor에 적용시켰을 때, amplitude 값은 오일러 phasor이 나타내는 원의 반지름과 같다. 즉 그 원의 직경은 amplitude의 2배이다.

옥타브 음을 만들려면 frequency의 배음들을 찾으면 된다. 440hz는 라(A)음인데, 여기서 이것의 배음들이라고 할 수 있는 880hz, 1760hz이나 220hz, 110hz 등을 사용하면 똑 같은 라(A)음을 나타낼 수 있다.

같은 theta 값이 들어간 코사인 함수나 사인 함수를 이용하여 소리를 만들면, 같은 소리가 들린다. 사인 함수와 코사인 함수의 차이는 파이/2라고 할 수 있는데, 이러한 phase의 차이를 인간은 감지하지 못하기 때문이다. 같은 shape의 함수라면 그것이 몇 radian을 이동했든 상관없이 모두 같은 소리가 들린다. 인간이 감지할 수 있는 차이는 frequency의 차이이지 phase의 차이는 아니다.

For loop를 이용해서 pulse train을 만들 수 있다. F0부터 Fend까지를 For loop에 대입하면서 spectrum 상에 보이는 frequency 그래프가 harmonics를 이루게 만드는 것이다. 이렇게 해서 만든 소리는 귀에 거슬리는 소리가 난다. 왜냐하면 이는 gradually decreasing이 아니고 산맥도 없기 때문이다. 따라서 gradually decreasing하게 만들고 산맥도 만들면 사람 목소리에 가깝게 만들 수 있다.

Gradually decreasing과 산맥을 만들려면 hz2w 함수와 resonance 함수를 사용해야 한다. 이때 hz2w 함수는 resonance 함수를 만들 때 쓰이고 우리가 직접 쓸 필요는 없다. 즉 우리는 resonance 함수만을 사용해서 gradually decreasing과 산맥을 만들 수 있다. Resonance 함수에는 RG라는 값이 쓰이는데, 이는 산맥의 중심의 위치를 일컫는다. 이때 RG는 x축인 frequency중에서 결정된다. BWG는 산맥의 뚱뚱한 정도이다. BWG가 높다는 것은 산맥이 뚱뚱해서 완만하게 간다는 뜻이고, BWG가 적다는 것은 산맥이 뾰족해서 경사가 급하다는 뜻이다. 이 RG와 BWG의 조정을 통해 gradually decreasing과 산맥을 나타낼 수 있게 된다. Gradually decreasing을 하려면 RG를 0으로해서 산맥의 중심이 0이라고 하여 다른 frequency 값들 전부 gradually decreasing되게 만들 수 있다.

데이터 – 기계 – 데이터

이때, 데이터는 벡터의 형태를 띄어야함. 여기서 기계는 함수 또는 인공지능임. 앞 데이터에 음성이 들어가서 텍스트가 나오면 음성인식. 앞 데이터에 텍스트가 들어가서 음성이 나오면 음성합성. 앞 데이터에 한국어 텍스트가 들어가서 일본어 텍스트가 나오면 기계번역. 모든 데이터는 벡터의 형태임. 그렇다면 기계의 형태는? 행렬의 형태이다.

입력의 벡터가 함수의 행렬을 통해 출력 벡터로 나타난다. 이것이 인공지능. 이때 그 행렬은 기계인데, 그 행렬의 값들은 많은 학습을 통해 행렬의 값들을 얻는 것. 그것이 기계학습. 즉, 인공지능은 행렬의 곱으로 입력 벡터를 출력 벡터로 바꾸어주는 것임. 그때 행렬을 거쳐야 하는데, 기계학습을 통해서 기계의 행렬값을 얻을 수 있음. 모든 데이터가 벡터의 형태를 띄어야하는 이유는 저런 행렬 곱셈을 하기 위해서임. 선형대수도 이와 비슷함. Linear algebra. 선형대수는 행렬을 가지고 하는 모든 이야기임. 그래서 모든 인공지능이 선형대수임.

행렬에는 column 벡터와 row 벡터가 있는데, 이들 각각을 따로 살펴볼 수 있다. 벡터들의 Linear combination으로 만들어지는 모든 가능한 값이 벡터 스페이스를 이룬다. Linear combination은 벡터에 스케일러를 곱하고 더하고 해서 만들어지는 값들을 일컫는다. 벡터 스페이스는 1차원의 모든 공간, 2차원의 모든 공간 등 어떤 차원의 모든 공간 자체가 벡터 스페이스가 되는 거지 어떤 차원의 일부분만 벡터 스페이스가 되는 것은 아니다.

Column의 관점에서 보자면, 어떤 행렬의 column 벡터들이 이루는 것을 spanning 해서 만들어지는 것이 column space다. 또는 column 벡터들의 linear combination으로 만들어내는 모든 공간이 column space다. Column space는 벡터 자체가 이루는 공간인 whole space를 넘어서지 못한다. 이때, column space의 차원은 independent한 column 벡터의 개수로 결정된다. 이는 row space의 차원과도 일치하며, rank라고 부른다. 즉 Independent한 column 벡터 또는 row 벡터의 개수를 rank라고 부르는 것. 이 때 independent하다는 것은 dependent의 반대말인데, dependent하다는 것은 같은 선상에 놓였다는 것은 물론, 어떤 두 벡터들의 linear combination으로 만들어질 수 있다면, dependent하다고 한다. Whole space의 차원에서 column space의 차원을 빼고 남는게 있다면 그것은 null space이다. Null space는 column space에 orthogonal하고 원점을 지나는 것이라고 이해할 수 있다. 또는 수학적으로는 xA=0 이 되는 모든 벡터나 행렬 등을 이른다.

위에서 column space의 차원과 row space의 차원이 같다고 한 것은 transpose의 개념과도 관련이 있음. 둘은 transpose된 관계에 있기 때문임. Transpose된 것은 차원이 무조건 같다.

Null space를 계산하는 데 있어서 행렬의 계산 상 그 벡터가 왼쪽이나 오른쪽에 붙을 때가 있다. 이때 왼쪽에 붙으면 left null space, 오른쪽에 붙으면 (right) null space라고 부른다. Column space의 null space를 left null space라고 부르고 row space의 null space를 (right) null space라고 부른다.

입력 벡터와 출력 벡터는 차원이 같을 필요 없고, 변화할 수 있는데, 이는 transformation matrix를 통과하면서 결정됨. 이때 transformation matrix의 기하학적 의미는 grid를 재설정하고 그 grid에 맞게 입력 벡터를 맞추면서 나타난다. 그런데 transformation에서 두 column 벡터가 dependent 하면 grid가 없이 선으로 눌린다. 이렇게 되면 determinant가 0이 되어 역함수를 만들 수 없게 된다. Determinant는 기하학적으로 grid의 한 칸의 넓이이다.

Eigenvector는 transformation matrix를 거친 입력벡터가 출력벡터로 나왔는데, 그것이 원점과 입력벡터를 이은 직선과 동일선상에 있는 것을 말한다. 그것이 기하학적 해석이다. 이 때, 이 transformation matrix의 Eigenvector을 물어보면 이 입력벡터를 말하면 된다. Eigenvector는 하나의 값만 일컫는 것이 아니라, 그 직선 위에 있는 모든 점을 말한다. 이를 Eigenvector space라고 한다. Eigenvector은 2x2 matrix에서 2개가 나온다.

Eigenvalue는 eigenvector에서 그것이 처음보다 확장되는 비율이다. 이는 기호로 λ(람다)로 나타낸다. Eigenvector의 수학적인 해석으로는 Av= λv이다. 여기서 λ는 상수이다. 이 수식의 의미는, v라는 입력벡터가 Av라는 출력벡터로 나왔을 때, 그것이 v에 상수 λ만 곱해져 처음과 동일한 직선 위에 위치해 있다는 의미이다.

Null space의 실용적, 응용적 해석.- 우리의 삶은 null space를 늘려나가는 과정. 어떤 일에 대해 숙련되면서 쓸데없는 동작 같이 보이는 것들도 그것이 숙련이 된다면 자연스럽게 할 수 있게 된다. stiff해지지 않고 더 자유로워진다는 뜻. 그래도 결과값은 항상 일정하게 나오게 만든다. Null space는 진화하면서 계속 적용되어 옴. 출력값을 하나의 목표로 봤을 때, 이것을 이루기 위한 여러 방법과 방해가 있어도 이를 피하고 목표를 이룰 방법 등을 만들어나가는 것 이것이 null space를 확보하는 것이다. 이것과 인공지능의 관련성은 예를 들어 강아지 사진이나 종류가 변해도 모두 강아지라고 인식할 때, 강아지 사진이나 종류는 null space 상의 변화일 뿐, 다른 곳으로 가는 변화가 아닌 것.

입력이 변하더라도 그것이 null space의 방향으로, null space과 평행하게 움직인 것이라고 한다면 출력은 항상 같다. 출력값이 꼭 0이 아니더라도 출력값이 변하지 않음.

Eigenvector는 어디에 쓰느냐? Column 벡터 2개가 있으면 Eigenvector 2개가 나옴 -> 2개의 벡터를 또 다른 2개의 벡터로 바꾼 것이라고 생각할 수 있음. 왜 이렇게 하느냐? Eigenvector를 우리말로 해석을 하자면 고유값 정도. 유니크한 것. Eigenvector로 바꾸면 그 고유하고 유니크한 것으로 바꿔준다. 예를 들어 국어, 영어는 언어적 능력이라 하고 수학, 과학을 수리적 능력이라고 하면, 언어적 능력과 수리적 능력을 딱 보여주는 것이 Eigenvector.

상관관계. 상관관계는 r로 표현 가능. -1 <= r <= 1. R이 0일 때 상관관계가 가장 작다. -1이면 음의 상관관계, 1이면 양의 상관관계. 절댓값이 1에 가까울수록 한 선상에 가까워짐.

두 벡터가 있을 때, 두 벡터 사이의 각도를 세타라 할 때, cos(세타)=r.

Inner product(dot product). 두 벡터가 있으면 안으로 같이 곱하고 더한 것. 이것의 기하학적 해석은 project된 변의 길이 x 벡터의 길이. 즉, a벡터의 길이 x cos세타 x b벡터의 길이.

Inner product를 써서 어떤 웨이브에서 어떤 주파수 성분이 많은지 가려낼 수 있다. 원래 웨이브와 다른 웨이브 여러 개를 inner product시켜서 그 값이 높을수록 그 주파수 성분이 많은 것이다.

두 벡터가 주어졌으면 a벡터의 길이, b벡터의 길이, inner product, cos세타, project된 변 길이 등 모두 다 구할 수 있다. 이때, cos세타는 cosine similarity라고 부르며, 상관관계와 아주 연관이 깊다. 이를 계산하는 법은 알아둬야 한다.

주파수 성분을 알아내기 위해 inner product를 할 때는 sin웨이브나 cos웨이브를 사용하면 phase shift에 민감하기 때문에, complex phasor을 써서 inner product를 하면 된다.

a벡터 길이, b벡터 길이가 같다고 가정하면, 세타값이 0에 가까울수록 inner product가 커지고, 세타값이 90도에 가까울수록 inner product가 작아진다.

Complex phasor로 inner product하면 complex number이 나오게 된다. 그래서 plotting이 불가능하게 됨. 그러면 어떻게 plotting 하냐? 절댓값을 씌운다. 절댓값을 씌우면 실수값이 나온다. | a+bi | -> 절댓값. 이 절댓값을 어떻게 해석하냐면, 실수를 x축, 허수를 y축으로 해서 그 점을 찍었을 때, 원점부터 그 점까지의 거리가 절댓값임. 이것이 complex number 절댓값의 기하학적인 개념. 이 절댓값을 활용해 inner product가 큰지 작은지 판단하고, plotting도 할 수 있는 것.

**뜻은 제곱으로 올리는 것. 뒷부분이 앞부분의 제곱으로 올라감. wi는 이미 지수로 올라갔으므로, 지수를 곱해주면 되는 것. 이때, 이 지수를 잘 정리하면, n[0...2파이]i 이런 형식으로 만들 수 있음. 그래서 대괄호 안이 세타 한바퀴 도는 것임. n에 1들어가면 한바퀴 돌고, n에 2들어가면 두바퀴 돎. 한바퀴 돌때나 두바퀴 돌때나 모두 100개의 샘플로 나타냄(샘플의 개수를 100개로 가정). 그래서 총 100바퀴까지 감. 이 100개 각각의 complex phasor을 모두 s와 dot product하는 것. complex phasor이 한 바퀴 돌면 100hz의 frequency에 해당하여 그것을 s와 dot product 하는 것, 두바퀴 돌면 200hz의 frequency에 해당하여 그것을 s와 dot product하는 것 이렇게 생각하면 됨. s와 z를 dot product. abs는 절댓값임. dot product를 해서 complex number이 나와도 절댓값으로 계산가능. 그리고 계산 할 때마다 그 값을 amp라는 빈 variable에 append하는 것. for loop이 끝나고 나서 len(amp)는? loop를 하는 횟수. amp에는 허수가 들어가지 않음. 절댓값을 취했기 때문. 이제 amp를 plotting 하면 끝.

Win size 초의 웨이브를 쪼개서 이만큼의 벡터들을 가지고 dot product 시킴. 그래서 그걸로 spectrum을 만듦.

Win step은 win size 만큼을 쪼갠 후, 그 win step초만큼 옆으로 이동해 다시 win size만큼 쪼개고, 그것을 반복하는 것. 그래서 같은 부분이 여러 번 중복되게 됨.

Spectrogram을 처음에 만들면 주파수가 높은 부분은 너무 희미하게 나옴. 그래서 한 번 더 처리를 가해야함. 이 때, dot product해서 절댓값 취한 결과가 1보다 크면 진하고 1보다 작으면 연함. 거기서 power spectrum을 하면 됨. 제곱을 해서 큰 값은 더욱 크게, 작은 값은 더욱 작게 함. 이렇게 하는 이유는 로그를 취하기 위함. 로그를 취하는 이유는 아무리 작거나 커도 로그를 취하면 일정한 간격으로 만들 수 있음. 너무 크거나 너무 작은 수들을 우리가 다룰 수 있는 범위 내로 바꿈. 로그를 취하면 그 값이 나올텐데, 이는 쉽게 다룰 수 있는 값들임. -4~4 이 정도. 여기서 대충 10 정도 더하면 이렇게 적당히 진하게 골고루 나옴.