데이터: 반드시 숫자화 되어야 한다.
데이터 종류: 소리, 영상, 텍스트, 숫자 데이터
벡터: 숫자의 열을 벡터라고 한다. (행렬과 달리 한 줄에 길게 쭉 나열해놓은 것.)
사진을 확대해서 보이는 하나의 픽셀은 각각 하나의 숫자다.
행렬: 숫자를 직사각형 안에 넣어놓은 것. 사진은 행렬의 형태다.
모든 데이터는 행렬으로 벡터로 전환할 수 있다. 사각형 안에 넣을지, 한 줄로 할지 선택하면 되고, 데이터는 일반적으로 벡터일 때 더 다루기 쉽다.

영상: 기본적으로 2차원. 여기에 컬러 이미지면 3차원, 여기에 시간까지 도입하면 4차원.
소리: 벡터화 할 수 있다. waveform의 소리를 한 줄로 나열하면 그것이 벡터화다.
텍스트: 50000개의 단어가 있다고 가정하면, 첫 단어를 10000...000, 두 번째 단어를 0100.....000, 마지막 단어를 0000...00001의 방식으로 벡터화할 수 있다.

***라이브러리**
파이썬에서 중요한 함수들을 정리해놓은 집합. 하나의 라이브러리 아래에 하위의 라이브러리가 존재한다.(import로 불러와야한다.)
형식: import numpy(필수!!!) as(선택)
import numpy from a를 통해서 하위의 함수로 바로 접근 가능 **시험!! 문법!**
ex. numpy안의, a안의, d 라이브러리의 함수: numpy.a.d.함수
***numpy란**
a = [1, 3, 5]
b = [2, 4, 6]
c = a+b
c [1, 3, 5, 2, 4, 6] 즉, 수학적 처리(사칙연산)가 되지 않고 단순 나열 형태로만 나옴
이때, 사칙연산이 가능하게 하는 것이 numpy임.
형식: import numpy (필수!!!)
A = numpy.array(a); B = numpy.array(b); A+B
import numpy as np (numpy를 np로 줄일 수 있음)
type(A) numpy.ndarray

X=np.array([[1,2,3],[4,5,6]])
x

11/19 ai와 데이터
음성의 데이터 형태들: 음성(벡터), text(벡터)
음성을 입력하여 text출력: 음성인식
입력, 출력되는 데이터는 모두 벡터 형태다. 그 중간과정에서 기계(인공지능, 또는 함수)는 행렬 형태로 데이터를 처리한다.
그렇다면, 그 벡터를 행렬로 처리하는 과정을 배우자.
인공지능 또는 기계 또는 함수: 행렬의 곱셈과정을 통해, 입력 벡터를 출력 벡터로 바꾸어주는 함수 역할. ---> “선형대수”

선형대수: Linear Algebra
행렬: Matrices: 행렬의 크기 = 행렬의 차원 = dimension
vectors: 1행의 행렬 or 1열의 행렬(한 줄) = a sequence of numbers
column matrices(열 행렬): 1열의 행렬(x,y 좌표 상에 점으로 표현 가능)
“행렬의 곱을 수치로 계산할 수 있는데, 동시에 그림에도 표현 가능.”(vector multiplication, addition 참고)

*******linear combination******
: 곱셈과 덧셈으로 이루어진 식(제곱 등등 없이)-> cv+dw의 형태
vector space: 여러 벡터들이 만들어낸 공간. 무한한 가능한 벡터(점)들로 채워진 공간
R: 1차원 공간 R의 제곱: 2차원(2행 1열) R의 n제곱: n차원 공간.(n행 1열 데이터 기반)
A = 2 -1 2 -1
    1 3을 1 3으로 나눠 무한의 linear combinations을 하면, 결국, 2차원 공간을 만들 수 있다. 즉, 3차원까지 확장되진 않는다.
slide29: col1과 col2는 다른 선상에 있으므로 서로 independent 하다고 한다.(원점과 col1,2는 삼각형 이룸.) col1, 2는 linear combination을 통해 각각의 선을 무한히 채울 수 있다.
slide37의 col1, 2: dependent (동일 선상).-> whole space: R의 제곱 Column space: L(1차원)
즉 column space가 independent하면 2차원, dependent하면 1차원이라고 볼 수 있다.

slide40의 3x3행렬: column space: R세제곱
41: col1,2가 서로 곱셈 관계이므로 P(plain, 2차원)
**42: 1*col1+ 1*col2 = col3인 경우.-> P 43: L(line, 1차원)
44: 열 행렬이 3개라서 전체 공간은 3차원, 하지만 점이 두 개 이므로 2차원을 못 벗어남.
44->45: Transpose: 3*2행렬의 데이터를 그대로 2*3행렬로 옮기는 것.
44와 45의 차이: whole space, column space는 항상 2차원이다!!!
m*n행렬은 두 개의 whole space: “R의 m제곱, R의 n제곱”을 가진다.(row/column space)
49: column space는 3차원, row space는 1차원(dependent)이므로, 그 사이의 공간이 남는데, 그 나머지 빈 공간을 통칭하여 “Null Space”라고 부른다.
50: Null space의 수학적 정의. 이때의 x1 x2를 null space
55: 총 정리

****Linear transformation***
Ax = b -> x가 입력벡터, b가 출력벡터. 입력벡터와 출력벡터의 차원이 꼭 같을 필요 x
ex) 1*4 행렬 곱하기 4*3 행렬: 에서 출력 벡터는 3차원(뒤 행렬의 열 수)
A: Transformation Matrices
62-65: 61의 계산식을 기하학적으로 해석한 것. -> transformation matrices의 역할
78 inverse matrix: 역함수를 취한 것****example 3: dependent한 경우!!! 주목!!
역함수를 취할 경우, independent할 경우, transformation matrix로 돌아갈 수 있으나, dependent할 경우 특정할 수 없다. 즉 invertible하지 않다!

*11/21
3*2행렬에 곱할 수 있는 행렬: 2*x. 즉 앞 열과 뒷 행이 같아야 함. 결과: 3*x 행렬.
Transpose도 해보고,
whole space/null space/column space: dimension 구분!
spanning: 두 independent column vector를 원점과 연결하여 column space를 만드는 것 (기하학적 해석) = linear combination
null space: column space와 수직인 것. = whole space - column space
spanning의 한계: spanning은 절대 whole space의 dimension을 넘어갈 수 없다.
*(1,2) (-1,0) (3,5) 세 개는 상호 independent할까? -> 아니다.
-> a*x1 + bx2 = x3로 표현 가능하면 independent 아님.
->기하학적 해석: whole space - column space = 0이므로, null space가 없다.
하나의 벡터는 몇 차원이든지에 상관없이 무조건 하나의 점이다.
requirements to become a vector space: linear combinations still stay in the space
R의 n제곱 space consists of all vectors with n components.
columns of A의 m*n제곱 in R의 m제곱 all their linear combinations form a subspace: column space, C(A)
column space dimension=row space dimension -> rank도 동일=independent columns
dim(whole space) n rows dim(column space) n of independent columns
***시험: 41의 column space가 plain인 이유: (1,2,4)*2=(2,4,8) (slide 41~45)**
orthogonal
50: (right) null space 52: left null space

58: Ax = b 행렬: 대문자 벡터: 소문자 x: 입력벡터 b: 출력벡터. x의 차원과 b의 차원은 반드시 같을 필요는 없고, 차원은 A에 의해 정의된다. (59 ~ 77: linear transformation 연습)
78 ~ 97: detransformation, 역함수, 역행렬 연습. **97이 중요! (cf) 행렬의 determinant)**

*eigenvector: 어떤 행렬의 eigenvector은 무엇인가?: 어떤 행렬의 eigenvector은 transformation의 결과 값이 원점과 같은 선상에 놓인다. 2차원 상에서 두 선, 3차원은 세 방향 존재. (밑 사이트에서 실습)

***null space의 수학적 해석: Ax = 0일 때의 x가 null space. x에 어떤 값을 입력하면 출력 0**
에 영향을 미치고, 또 다른 값들을 입력하면 출력에 영향을 미치지 않는다. 이때, 후자의 값들을 null space라 함. null space는 인공지능에서 굉장히 중요하다. 예를 들어, 인공지능 기술로 강아지 사진을 찾아낸다고 할 때, 강아지의 사진은 무한한데, 어떠한 변화를 줘도 강아지의 범주 안에 포함되는 경우는 전자, 강아지의 범주를 벗어나게 하는 변화들을 null space라고 본다.
http://setosa.io/ev/eigenvectors-and-eigenvalues/: 행렬의 입력, 출력 값 실습 가능.
벡터란 방향이다. 방향이 중요하다. 선상, 공간상 위치는 중요하지 않다. 방향이 중요.

eigenvectors/eigenvalues의 의미: 고유 벡터. 2차원 상에서 어떤 두 벡터를 또 다른 두 벡

터로 바꾸는 것.

상관관계: correlation: x,y,z,.... 변수 사이의 값이 서로 영향, 연관이 있는 관계. ex) y=x
범위: -1<=r<=1. 이때 r=0이 최소의 값. 0미만 값은 음의 상관관계.
정확한 상관관계: r=1,-1 직선에 수렴. r=0에 가까울수록, 변수의 집합은 원에 가까워진다.
좌표 상의 임의의 두 점(벡터)과 원점 사이의 직선이, 원점과 이루는 중심각을 sector라고 할 때 둘 사이의 상관관계 r=cos(sector) 다. -1<=cosx=<1 이므로, 상관관계와 같은 범위다. 이번 학기에서 가장 중요한 내용!!
inner product=dot product: 두 벡터 (1,2,3), (4,5,6)을 안 쪽으로 곱한 것. 즉 (4,10,18) 이를 a.b로 표현할 수 있으므로 dot product라고도 함.
inner product의 기하학적 해석: a, b 사이의 수선의 발을 내린 것. 이때, 이 수선의 발, 원점과 a 사이의 직선, 원점과 b 사이의 직선은 합쳐서 직각 삼각형을 이룬다. inner product의 수식적 값은 lal*cos(sector)*lbl
2차원 상에서 r이 양수면, 1, 3사분면을 지나고, 음수면 2, 4 사분면을 지난다. | r | 이 1에 가까울수록 y축에 수렴한다.
****시험: 두 벡터가 같은 직선상에 있으면, 둘 사이의 각도는 180도 이므로 cos=-1=r이다. 이를 cos wave로 표현하면, 서로 180도 차이가 나므로 정반대의 y축 값을 가지는 두 wave로 표현된다. 이는 서로 음의 관계만 가지면 되지 amplitude와는 상관이 없다.(amplitude는 벡터의 길이와만 상관있지 벡터 간의 관계와는 상관없다.)**

11/28
eigenvector: Av=λ(상수)v=한 직선 상에 있다.
inner product: 원점 임의의 a벡터, b벡터 사이에는 삼각형을 이루는데, 이때 a*b의 곱셈 방법은 두 가지다. (1*3)*(3*1)=1*1: inner product, (3*1)*(1*3)=3*3: outer product. inner product의 결과 값은 항상 1*1행렬이다.
inner product의 기하학적 해석: 원점, a, b의 삼각형에서, a에서 수선의 발을 내려 b에 내린 수선의 발을 b`라고 하면, inner product=b`*b=lal*cos(sector)*lbl
a가 (1,2,3) b=(2,4,7)일 때, lal= √1제곱+ 2제곱+ 3제곱 lal*lbl=cos(sector)
****시험: 두 개의 벡터가 주어졌을 때 cos similarity를 구하라**
임의의 complex tone sine wave가 100개의 벡터 값으로 이뤄졌다고 할 때, 그 complex tone을 두 개의 pure tone으로 나눠보자. 이 때 두 pure tone도 각각 100개의 벡터 값으로 구성될 것이다. 그리고 다시 각각의 pure tone으로 다 분해해보자(phasal)
이때 sine wave a,b,의 주파수가 100hz로 같고, c의 주파수가 200hz일 때, 각각을 inner product하면 a*b>a*c (c에서는 감소하는 부분이 많기 때문.)
sin wave* cos wave를 그대로 inner product하면 0이 나온다. 이때 둘을 벡터로 표현하여 sector를 구하면 90도가 나온다.

inner product=b'*b=a'*a=lal*lbl*cos
어떤 complex tone의 spectrum에서 각 frequency(100hz, 200hz,...1000hz...)의 pure tone의 구성비율을 알아내는 작업을 할 것이다. 이때 사용되는 것이 inner product다.
각 pure tone은 각각의 벡터다.

sin cos wave는 phaser에 대한 민감도가 너무 좋지 않다. 즉 위의 sin cos의 사례처럼 90도 이동하면 모든 값이 0이 나오는 불상사가 나오기도 한다. 따라서 쓸 수 없다.
대신에 complex phaser를 사용한다. 이때 complex phaser의 벡터 수는 구하고자하는 complex tone의 벡터 수와 같아서 서로 대응돼야 한다.
complex phaser는 a+bi의 허수 형태까지 가능하다. 이때 사용하는 것은 a+bi의 절댓값이다. 이는 기하학적으로 허수의 좌표 상에서(complex plain), 원점과 (a,b) 사이의 거리다.

spectrogram/spectrum 각 축 값들 알아두기.
fourier tranform: 스팩트럼.-> 여기에서 스팩트로그램을 만드는 법: 한 장 한 장의 스팩트럼을 0.xxxx초 단위로 만들어서 이어 붙이면 된다. 스팩트럼에서도 저주파 부분이 강하고 고주파 부분이 약했듯이, 스팩트로그램에도 똑같다. 아랫부분이 색이 진하고 위는 색이 연하다. 위쪽이 너무 희미한 기본적인 스팩트로그램에서 한 번 더 처리를 하면 아래쪽의 전반적으로 진한 색의 스팩트로그램이 완성된다.
처리과정: 103: magspec**2: 제곱을 해서, 1보다 큰 값은 훨씬 크게, 1보다 작은 값은 훨씬 작게 처리한다. 실제 밑에 그림을 보면 저주파는 더 까매지고 고주파는 더 연해졌다.
2. log를 취해주면 다시 값이 1이하의 값들이 커지면서 고주파 부분의 색깔이 진해진다.