모든 것은 데이터로 되어있다.

따라서, 모든 것은 벡터값으로 표현할 수 있다.

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}$$

행렬은 정의상 직사각형 형태로 되어있음

$$\begin{bmatrix} a_1 \\ \vdots \\ a_m \end{bmatrix} \begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix}$$

가로는 column이라고 부르고 세로는 row이라고 부른다.

이 행렬의 크기를 얘기할 때 demention(차원)의 이 차원은 공간으로 전환되고 벡터 공간는 m*n의 형태 벡터들이 될 수 있는 모든 값으로 이뤄져 있는 공간(= 차원 / N개의 componant가 그 차원을 다 채우면 그 차원의 vector space 하나를 이룬다.)을 뜻하며 이것이 바로 column 측면에서는 m차원 row 측면에서는 n차원이 된다. 즉, 하나의 행렬은 m이라는 차원과 n이라는 차원을 whole space로 가진다.

공간은 whole space와 column/row space으로 나눌 수 있는데 whole space 란 이 벡터 자체가 가지고 있는 차원 전체를 뜻하고 column/row space는 이 column 혹은 row가 채울 수 있는 차원 전체를 뜻한다.

또한 null space가 있는데 whole space에서 어떤 column/row space를 정의하고 나서 그 나머지를 null space라고 한다.<기하학적 정의>

어떤 행렬이 있을 때 무엇을 곱하든 0으로 이루어진 행렬이 될 때, 이 모든 값들이 이루는 공간을 null space이라고 한다. <수학적 정의>

-> column서는 왼쪽에다 곱해야지 차원이 같아지기 때문에 left null space라고 한다. Row의 입장에서는 오른쪽에서 곱해야지 차원이 같아지기 때문에 null space라고 한다. 그리고 하나의 행렬에서 column입장에서 null space를 빼고 row입장에서 null space를 빼면 둘다 같은 차원을 가진다. 즉, whole space는 달라도 둘의 차원은 같다는 뜻이다.

Cf)

[영공간

Ax=0의 모든 해 집합을 행렬 A의 **영공간(null space)**이라고 합니다. A가 n개 열을 가졌다면 A의 영공간은 Rn의 부분공간이 됩니다.

영공간을 **선형변환(linear transformation)** 관점에서 이해할 수도 있습니다. T를 n차원 벡터 x를 m차원 영벡터로 변환하는 선형변환으로 둔다면 영공간 NulA는 아래 그림처럼 도식화할 수 있습니다.]

M*N인 행렬이 있을때

M은 column의 관점 / column space

-> M차원이 column의 whole space

만약에 column space가 whole space를 다 채우지 못한다면 이는 null space라고 한다.

<Null space는 그 벡터 값에 곱하면 0을 만들어주는 행렬>

또한 모든 column이 독립적이면 (한 일직선상에 존재하는 행렬이 하나식만 존재) column space는 whole space다.

여기서 독립적인 행렬은 whole space가 m차원이면 최대 m개까지만 존재가능하다.

-> m+1차원 행렬은 m차원의 행렬의 조합으로 표현가능하다.

만약에 column vector가 두개가 있다면 행렬의 덧셈으로 그 vector 값을 구할 수 있다.

이는 곧 두 vector에서의 평행사변형 꼴의 한 점이 된다.

Vector가 한 직선상으로 되어있지 않으면 두개의 rank으로 되어있는 행렬이다.

N은 row의 관점 -> N차원이 row의 whole space

2차원 row가 3개가 있다고 하면 2개의 조합으로 나머지 하나를 만들 수 있기 때문에 3개 전부 independent할 수 없다. 결국, n차원에서 independent한 벡터는 총 n개만 존재한다.

요약)

R^m –> m이 column의 whole space-> column space

R^n -> n이 row의 whole space -> row space

-> 그리고 둘의 차원은 같아야 한다. (whole space는 달라도)

그리고 이 모든 벡터 값은 이를 좌표축 상에 표시가 가능하다. 여기서 벡터는 크기와 방향성만 중요하지 이 벡터가 꼭 3,4의 위치에 있어야 하는 것은 아니다.

크기와 방향성만 정해지면 어디로 가든지 상관이 없다.

즉, 원점에서 시작할 필요가 없다는 뜻이다.

또한, 벡터 크기는 차원을 늘릴 뿐 점을 늘리지는 않는

다. 반대로, 벡터에 곱해진 값(상수)은 벡터의 길이에만 영향을 주지 크기에 영향을 주지 않는다. 이를 Spaning이라고 한다. 예를 들어 column 벡터 두개에서 이를 확장하면 하나의 삼각형이 만들어지고 이 삼각형이 무수히 확장되면 column vector space를 만들 수 있다.

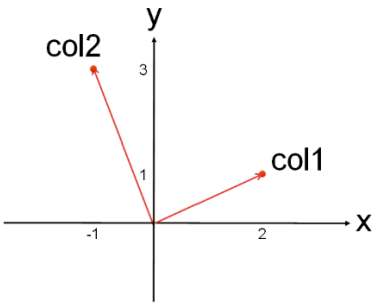
-> 상수의 곱셈은 벡터의 길이를 / 두 행렬의 덧셈은 (평행사변형 형태로)새로운 방향으로 바꿔준다. /두 행렬의 곱셈은 벡터 값의 차원을 틀어준다.

두개의 벡터값은 결합이 가능한데 이를 linear combination이라고 하고 이는 선형성(같은 직선위에 존재하는 여부 / 독립성 여부)를 판단하는 기준이 된다.

$$c \cdot v + d \cdot w$$

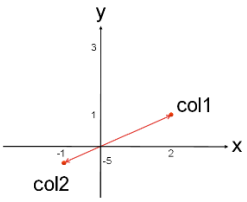
V는 벡터 W도 벡터(차원이 같음) / c,d는 곱하는 숫자

Linear combination은 이 형태만 가능



이 두개는 independent하다고 한다. -> 같은 선상에 없기 때문.

그래서 이 두개의 벡터의 결합으로 다른 점을 나타낼 수 있다. = col1과 col2의 linear combination을 통해



이렇게 한 직선상에 있으면 dependent하다고 한다.

-> 삼각형이 만들어 지지 않는다. -> 여기서 확장을 해도 line이 확장될 뿐 line 안쪽을 채울 수 없음

이를 독립성을 행렬식으로 표현하면

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 2 & 1 & 3 \\ 4 & 1 & 5 \end{bmatrix}$$

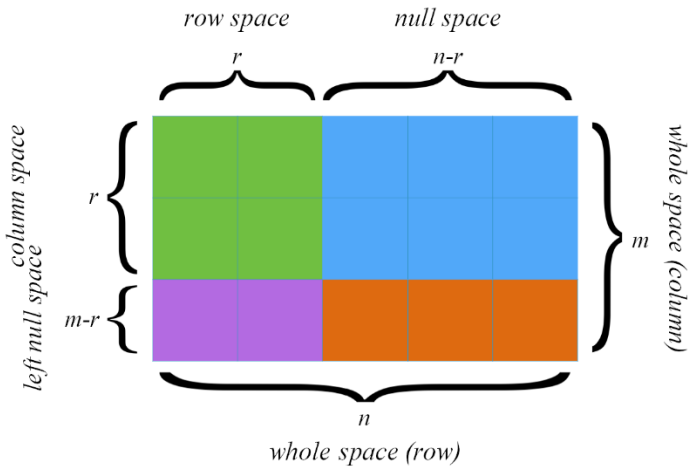
Column space -> 독립적이 아니다. -> linear combination으로 만들 수 있으면 이는 independent하다고 할 수 없음. -> 1,2,4 / 1,1,1 이 두개에 의해 2,3,5를 만들 수 있기 때문에 2,3,5는 존재하지 않는다고 할 수 있음(만들어 지는 것이지 새로운 값이 아니다.) 결국 whole space는 3차원이지만 column space는 2차원이 된다.

$$A^T = \begin{bmatrix} 1 & 2 & 4 \\ 3 & 3 & 1 \end{bmatrix}$$

T는 transpose했다는 뜻이다. -> 3*2 -> 2*3로 바꿨다 -> whole space는 2차원 <여기서 whole space는 열의 관점에서 봤을때의 whole space / 2차원 값이 벡터가 3개 있는 것>

Colunm vector는 3개가 있는데 만들 수 있는 것은 2차원이기 때문에 column space는 2차원이다.

Transpose를 해도 여전히 whole과 vector의 개수는 달라져도 column space는 2차원



전체의 요약

m*n 행렬이 있다고 생각할 때 m 크기의 whole space<column>가 하나 있고 n 크기의 whole space<row>가 있다. M이랑 같거나 작게 column space<column>가 있고 N이랑 같거나 작게 column space<row>가 또 존재 한다. 여기서 column space의 차원을 판단하는 기준이 독립적이냐 독립적이지 않느냐에 따라 판단한다. 여기서의 나머지를 left null space / null space라고 한다.

$$A\vec{x} = \vec{b}$$

X가 입력 벡터, b가 출력 벡터, A는 행렬

X의 벡터 차원과 b라는 출력 벡터의 차원은 같지 않아도 된다.

A 행렬이 그 역할을 해준다.

A라는 행렬은 x의 벡터 크기에 따라 크기가 정해지고

B라는 벡터는 A라는 행렬에 따라 크기가 정해진다.

-> A라는 행렬로 b라는 벡터가 달라지기 때문에, A를 linear transformation이라고 한다.

<차원도 바꾸고 값도 바꾸었다.>

이를 그래프 상으로 보면 basis vector의 차원을 틀어주는 것과 같다.

->

기하학적 해석: 선형변환은 공간을 이동시키는 방법이며 격자선이 여전히 평행하고 균등간격을

유지한 변형이다. 그리고 원점은 고정되있음을 의미하고 이 변환들을 간단한 숫자들로 설명가능하다. 바로 기저 벡터들는 변환후의 좌표값이다.

행렬은 우리에게 이러한 변환을 설명하는 언어를 제공해줌.

행렬의 열들은 이 좌표 값을 나타내며, 행렬=벡터 곱셈은 단지 이것을 계산하는 방법임.

이 변환이 주어진 벡터에 적용한 결과를 보여줌.

우리가 행렬을 볼때마다 공간의 어떤 변환으로 생각가능.

$$A^{-1}b = x$$

역함수의 형태

출력에 해당되는 부분에 A의 역행렬을 집어넣어서 입력 값을 유도하는 과정

하지만, 만약에 A로 인하며 b가 일직선이 되면

여기 있는 점을 역으로 했을 때 이 점은 원래 자리로 찾아 갈 수 없음

직선에서 펼치려고 하면 어디서 왔는지, 어디로 가는 지 알 수 없음 -> 이를 invertible하다고 한다.

행렬을 배운 사람 determinant<선형대수학에서, **행렬식**은 정사각행렬에 수를 대응시키는 함수의 하나이다. 대략, 정사각행렬이 나타내는 선형 변환이 부피를 확대시키는 정도를 나타낸다.> 이 값이 0이라고 할 때 역행렬이 존재하지 않음

면적이 0일 된다면(독립적이면) -> 0이 됨과 동일해지고 왜 inverse가 존재하지 않느냐를 설명해준다.

어떤 행렬의 eigenvector란 무엇이냐?

어떤 벡터는 행렬 A와 transformation을 했음에도 불구하고 원점과의 일직선상에 있는 경우가 존재하는데 이 벡터를 eigenvector라고 한다.

행렬 A를 선형변환으로 봤을 때, 선형변환 A에 의한 변환 결과가 자기 자신의 상수배가 되는 0이 아닌 벡터를 고유벡터(eigenvector)라 하고 이 상수배 값을 고유값(eigenvalue)라 한다.

1. Null space

null space가 필요한 이유
2*3 3*1 = 2*1
whole space는 3차원
row space는 2차원 (서로서로 independent 한 상황 -> 2차원)
두 개의 row vector
여기서 null space는 1차원(=row space의 평면을 직각으로 지나가는 직선 하나, 평행하게 따라가는 직선 / 이 방향으로의 변환은 출력에 영향을 미치지 않는다.)
출력에 영향을 미치는 공간은 row space / 영향을 미치지 않는 공간 null space

1) 기하학적 해석

위에 설명했던 행렬로 설명을 하면, Colum space가 있는 전체 집합은 3차원이다.
Row space는 row들이 span 하는 subspace를 의미하며, 기본적으로 2차원에 포함된다.

- Nul $A = (\text{Row } A)^{\perp}$.
- Nul $A^T = (\text{Col } A)^{\perp}$.

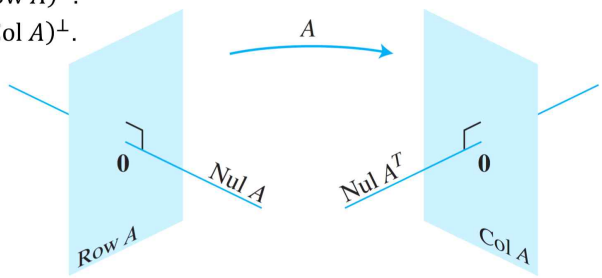


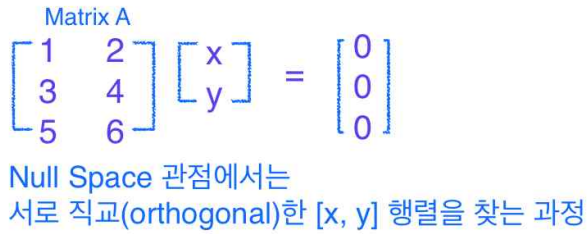
FIGURE 8 The fundamental subspaces determined by an $m \times n$ matrix A.

선형 독립하는 벡터를 Gram Schmidt를 통해서 각도가 1도라도 삐져나온다면, projection을 하여 수직은 벡터를 찾을 수 있다. 즉, 선형 독립(linearly independent)하는 벡터를 찾을 수 있다면 Gram Schmidt를 하여 직교하는 벡터를 찾을 수 있다.

각도가 선형 종속(linearly dependent)이라면 평면 안에 들어온다는 뜻이다. Projection 하면 평면과 벡터의 각도는 '0'이 된다. 1도만 있어도 선형 독립(linearly independent)하게 된다. 직교(orthogonal)한다는 그것은 이 각도가 90도가 된다. 따라서, 선형 독립이 직교하지 않을 수 있지만 직교한 벡터는 항상 선형 독립이다.'

Row A 벡터 [1, 2, 3]과 선형 독립인 벡터를 직교하는 벡터들의 집합은 평면을 이루므로, Row space가 선이라면, Null space는 평면이 된다. 따라서 Null A 또는 base 벡터는 2차원이 된다….

2) 수학적 해석



Ax=0이라고 정의 (null space)
given A에 대해서 0(영행렬)이 되는 x값을 찾아라 -> x값의 공간이 null space
= 내적 관점에서는 '0'을 만족하는 x, y를 구하는 것이나, 서로 직교한다면 내적이 '0'이 되는 개념을 활용한다면, 매트릭스 A 의 각각의 row 벡터(a1, a2, A3)와 직교하는 [x, y] 벡터의 좌표를 구하는 것과 같다.

3) 응용의 해석

결국, null spaces는 어떤 입력이 들어오든 출력에 영향을 미치지 않은 공간 -> 입력이 변하더라도 출력에 영향을 미치지 않은 공간을 줄이면서 필요 없는 정보를 점점 줄일 수 있음. -> 변환해서 최대한 간단히 해서 줄이는 것이 중요
그래서, null space와 그렇지 않은 공간을 나누는 것이 핵심 / 방해받는 공간을 피할 수 있게 해줌 = 입력이 많이 변해도 -> 출력의 영향 여부로 판단 -> 하지만 수학적 해석처럼 꼭 출력 값이 0일 필요는 없음

2. eigenvector(고유벡터)와 eigenvalue(고유값)

1) 기본개념

(1,0)

(0,1) -> 어떤 값을 집어넣어도 그대로 되는 행렬 -> basis vector라고도 한다. 이 상태에서 grid를 바꿔서 값을 변경 가능

V는 원래 벡터값 -> 입력 / Av는 변환된 벡터값 -> 출력 / 변환된 값은 평행사변형의 꼭짓점으로 바뀐다.

-> 이게 기본적인 transformation

Av = λv // $(A - \lambda I)$ v = 0 (v -> 0을 포함하면 안 된다.)

v는 고유벡터 <eigenvector> / λ는 고유값(상수) <eigenvalue>이 된다. -> eigenvector의 방향성은 바뀌지 않고 크기만 바뀐다(vector 값의 확장이 된다) -> 모든 벡터값은 한 직선상에 있다.

-> 결국 고유벡터를 통해서 입력벡터을 만들어 주어 계산을 간소화한다. -> 행렬에서의 계산에서 상수로의 계산으로 대체되기 때문에 계산과정이 간소화된다..

원점과 입출력이 평행한 순간이 존재 -> eigenvector

eigenvalue -> 벡터값이 변하는 정도 / 크기 (= 직선일 때의 AV/V의 값이 eigenvalue이다)-> 2차원에서는 2개가 존재(eigenvector가 두 개 존재하기 때문)

이를 수식으로 표현하면 **Av=λv.**

v는 고유벡터 / λ는 고유값이 된다.

-> 결국, 입력 vector 값에서 방향성은 바뀌지 않고 크기만 바뀐다.

-> 행렬에서의 계산에서 상수로의 계산으로 대체되기 때문에 계산과정이 간소화된다..

- 고유벡터를 구하는 법 : $(A - \lambda I)\ \mathbf{x} = 0$
- I -> 단위행렬 (1,0)
  (0,1)
- x -> 0을 포함하면 안 된다.
- $A - \lambda I$는 x 와 선형 비독립적 행렬(선형 독립적인 행렬 A를 λ을 통해서 선형 비독립적으로 만들어 준다.)

2) 사이트에서 설명

To begin, let v be a vector (shown as a point) and A be a matrix with columns a1 and a2 (shown as arrows). If we multiply v by A, then A sends v to a new vector Av. If you can draw a line through the three points (0,0), **v and Av, then Av is just v multiplied by a number λ; that is, Av=λv.** In this case, we call λ an eigenvalue and v an eigenvector.

Those lines are eigenspaces, and each has an associated eigenvalue. Second, if you place v on an eigenspace (either s1 or s2、대칭하여 2개가 생김 양의 방향과 음의 방향) with associated **eigenvalue λ<1, then Av is closer to (0,0) than v**; but **when λ>1, it's**

farther. Third, **both eigenspaces depend on both columns of A**: it is not as though a1 only affects s1.

3) null space and eigenvector

벡터라는 것은 matrix의 한 형태라고 했지만, 물리학적인 개념으로 벡터는 방향이다.
null space도 방향이 중요 / eigenvector도 방향이 중요
eigenvector -> 어떤 행렬이 존재할 때 이 행렬의 eignvector는 무엇인가? 이러한 질문을 해야 한다. 원점에서의 직선 -> 이것의 방향 -> eigenvector는 2*2에서 두 개가 있다. 3*3에서는 3개가 있다.

4) eigenvector는 왜 배우는가?

column vector가 두 개 -> 이를 또 다른 두 개의 vector로 바꾼 것 -> eigenvector가 2개 존재하기 때문 -> 하**는 이유? 훨씬 더 본질적/ 고유한게 무엇인지 판단하게 해줌. 2개 중에 어떤 게 더 고유한가? -> 행렬 대신 λ로 계산 가능하게 해준다. -> 즉 계산을 빠르게 하기 위해, / 어떤 벡터든지 고유벡터(eigenvector) <Av = λv가 되게 하는 v>들의 선형결합으로 표현을 하고/ 각각 벡터와 그 값을 곱한 걸 나중에 합치는 과정(=eigen decomposition)을 살펴보고자 한다**. 통계학에서 PCA / 분산분석을 하게 해준다.

3. correlation and vector

1) 기본개념

상관관계 : 같이 가는 느낌 -> correlation

r의 값 : -1<=r=<1

-1와 1의 값이 나올려면 선상에 있으면 된다. -> 얼마나 선상에 가까우냐에 따라 -1,1 / 상관이 없을 때 -> r의 값이 0

벡터의 내적 (inner product)

여러 개의 수치가 있어도 두 변수만 있다면 이 모든 수치는 두 개의 벡터로 표현할 수 있고 이 두 벡터는 차원이 넓다고 해도 삼각형 형태를 벗어 나지는 않는다.

-> cos을 적용 가능

a = [1,2,3]

b = [2,4,7]

T -> 행과 열을 바꿔 준다.

a*b^T = (1*1) = 31 -> 값이 하나만 나오도록 -> 이석이 inner product(벡터의 내적)이라고 한다. -> $a \cdot b(\sum a_i * b_i)$라고 표현 가능

b벡터의 선(길이)에 project된 a벡터의 값(길이)를 곱하면 내적 값이 나온다.

이를 알려면 $a \cdot b = |a|*\cos(\theta)*|b|$를 구하면 된다.

<삼각형이기 때문에 a벡터의 project/|a벡터의 길이| = cos(theta)>

$a \cdot b/|a|*|b| = \cos(\theta)$

이는 correlation과 같다

결국 $a \cdot b$(inner product)/|a||b|=cos(θ)=r

(상관계수식으로 증명) -> 두 변수의 공분산 -> 벡터의 내적 / 두 변수 각각의 표준편차의 곱 -> 각각 벡터 길이의 곱

$-1 \leq \cos(\theta) = r \leq 1$

cos similarity -> 두 개의 벡터를 주고 이를 구해라 -> 얼마나 비슷한지를 구하는 법

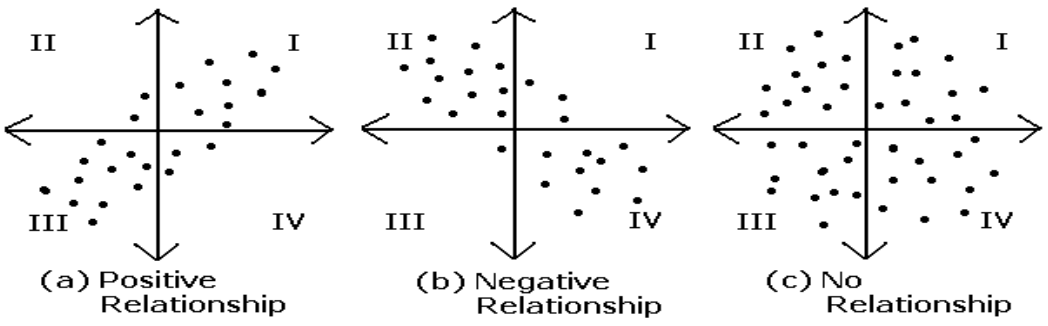
cf) a^T*b = (3*3) -> 행렬로 나오게 하는 방법 -> outer product

2) 통계학적 공분산과 상관계수

공분산(Covariance)이다.,

두 확률변수 X와 Y가 어떤 모양으로 퍼져있는지

즉, X가 커지면 Y도 커지거나 혹은 작아지거나 아니면 별 상관 없거나 등을 나타내어 주는 것이다.



Cov(X, Y) > 0 X가 증가 할 때 Y도 증가한다.
Cov(X, Y) < 0 X가 증가 할 때 Y는 감소한다.
Cov(X, Y) = 0 공분산이 0이라면 두 변수간에는 아무런 선형관계가 없으며 두 변수는 서로 독립적인 관계에 있음을 알 수 있다.

그러나 **두 변수가 독립적이라면 공분산은 0이 되지만, 공분산이 0이라고 해서 항상 독립적이라고 할 수 없다.**
어떻게 하면 그것을 나타낼 수 있을까 고민한 결과
공분산은 아래와 같이 구하기로 하였다.

확률변수 **X의 평균(기대값), Y의 평균**을 각각(설정)
$E(X) = \mu,\ E(Y) = \nu$
이라 했을 때, **X,Y의 공분산**은 아래와 같다.
$Cov(X, Y) = E((X - \mu)(Y - \nu))$
즉, 공분산은 **X의 편차와 Y의 편차를 곱한것의 평균**이라는 뜻이다.
$$Cov(X, Y) = E(XY) - \mu\nu$$
$$\because E((X - \mu)(Y - \nu)) = E(XY - \mu Y - \nu X + \mu\nu)$$
$$= E(XY) - \mu E(X) - \nu E(Y) + \mu\nu$$
$$= E(XY) - \mu\nu$$
그리고 **X와 Y가 독립이면**
$E(XY) = E(X)E(Y) = \mu\nu$ 이므로 **공분산은 0**이 된다.

그런데 **공분산에도 문제점**이 하나 있다.
X와 Y의 단위의 크기에 영향을 받는다는 것이다.
즉 다시말해 100점만점인 두과목의 점수 공분산은 별로 상관성이 부족하지만 100점만점이기 때문에 큰 값이 나오고
10점짜리 두과목의 점수 공분산은 상관성이 아주 높을지만 10점 만점이기 때문에 작은값이 나온다.

이것을 **보완**하기 위해 **상관계수(Correlation)**가 나타난다.
상관계수라는 개념이 왜 나왔는지 생각하다 보면 의외로 간단하다.
확률변수의 절대적 크기에 영향을 받지 않도록 단위화 시켰다고 생각하면 된다.
즉, **분산의 크기만큼 나누었다고 생각**하면 된다.
상관계수의 정의는 아래와 같다.
$$\rho = \frac{Cov(X, Y)}{\sqrt{Var(X)\ Var(Y)}}, \quad -1 \le \rho \le 1$$

상관계수의 성질을 나열해 보자
1. 상관계수의 절대값은 1을 넘을 수 없다.
2. 확률변수 X, Y가 독립이라면 상관계수는 0이다.
3. X와 Y가 선형적 관계라면 상관계수는 1 혹은 -1이다.

$$r = \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{x_i-\overline{X}}{s_{\overline{X}}}\right)\left(\frac{y_i-\overline{Y}}{s_{\overline{Y}}}\right)$$

$$= \frac{1}{n-1}\frac{\sum_{i=1}^{n}(x_i-\overline{X})(y_i-\overline{Y})}{\left\{\left(\frac{1}{n-1}\right)\sum_{i=1}^{n}(x_i-\overline{X})^2\right\}^{\frac{1}{2}}\left\{\left(\frac{1}{n-1}\right)\sum_{i=1}^{n}(y_i-\overline{Y})^2\right\}^{\frac{1}{2}}}$$

$$= \frac{\sum_{i=1}^{n}(x_i-\overline{X})(y_i-\overline{Y})}{\left\{\sum_{i=1}^{n}(x_i-\overline{X})^2\right\}^{\frac{1}{2}}\left\{\sum_{i=1}^{n}(y_i-\overline{Y})^2\right\}^{\frac{1}{2}}}$$

https://angeloyeo.github.io/2019/08/20/correlation_and_inner_product.html

3) 상관관계에서 선형대수가 쓰이는 방법

차원이 높아져도 삼각형을 이룬다.

여러 개의 벡터값이 직선을 이룬다.

-> **선형결합 vector의 개수는 변수의 개수지 변수 값의 개수가 아니게 된다.(하나의 변수는 하나의 vector로 모든 변수값을 나타낼 수 있다.)**

각도 값(θ)에 cos(θ)을 해주면 cor 값이 된다.

관련성이 없을수록 각도값이 커진다(최대 90도) -> cor값은 0

관련성이 있을수록 각도값이 작아진다.(최소 0도) -. cor값은 1 // 90도 초과해서는 -1에 가까워 진다.

어떤 두 벡터가 있을 때 A라는 벡터 B라는 벡터가 있을 때 그 사이의 각도값을 구하는 방법

-> **inner product : 안쪽으로 다 곱해서 더한 값**

ex) (1 2 3) a 벡터 (둘이 하나의 행렬을 이룸)

(4 5 6) b벡터 => inner product 값 : 32

= 하나를 수직으로 내려서 두 길이를 곱한 값 (기하학적)

a벡터의 길이 * cos(θ) * b벡터의 길이 = inner product

길이 구하는 방법 = 루트(1^2 +2^2 + 3^2) = a벡터의 길이 구하는 방법

inner product = →a · →b=|→a|*cos(θ)*|→b| <cos(θ) = 정사형의 길이 / →a>

상관계수 식에서 →a=xi－X(x의 표본평균), →b=yi－Y(Y의 표본평균)라고 하고 이를 계산하면

$$= \frac{\sum_{i=1}^{n} a_i b_i}{\left\{ \left(\sum_{i=1}^{n} a_i a_i \right) \times \left(\sum_{i=1}^{n} b_i b_i \right) \right\}^{\frac{1}{2}}}$$

결국
→a · →b(inner pro duct) / |→a||→b|=cos(θ)=r(상관계수식으로 증명)
－1≤ cos(θ) = r ≤1

$$= \frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|}$$

https://angeloyeo.github.io/2019/08/20/correlation_and_inner_product.html
상관계수와 공분산 모두 공통적으로 벡터의 내적을 이용해 설명할 수 있고, 데이터 간의 (즉, 벡터 사이의) '닮음'과 연결 지을 수 있다.
다만, 둘 사이의 가장 두드러지는 차이점은 정규화의 방법인데, 상관계수는 벡터의 크기로 정규화해주는 반면 공분산은 벡터의 원소 수를 가지고 정규화해준다.

4. 소리와 상관계수

1)기본개념

소리도 벡터 값 -> 이 벡터에다가 사인 웨이브를 여러개 만들어서 각각 inner product을 해버리면(|→a||→b|값은 고정 / 소리의 벡터 값도 고정되어있기 때문 / 지속시간이 존재) 어떤 값이 나올 텐데 데 서로서로 유사한 성분이 많이 있으면(correlation이 높으면) inner product 값이 높게 나올 것이다.

2) wave와 벡터

wave에 대한 설명
wave가 있다고 생각하면 그냥 눈으로 볼 때 여기에 어떤 성분들이 2개가 보이는가?
-> 사인 wave -> 1)주파수와 2)amplitude가 보인다.
주파수 -> 벡터값으로 표현 가능
두 개의 주파수의 inner product를 이용 가능
사인 웨이브를 만드는 것은 phasor라고 했음
만든 만큼 곱하면 그 숫자만큼 만들어짐
그 phasor를 이용하여 cos similarity를 이용하면 어떤 부분과 같은지가 구할 수 있다. ->즉, 어떤 성분 (사인 웨이브)가 많은지를 구해서 이를 종합하면 된다.

ex)
a라는 사인 웨이브
b라는 사인 웨이브 (a랑 비슷)
c라는 사인 웨이브(a,b보다 더 빠름)
a、b > a、c -> 같은 성분에만 response -> 같은 시간 상에서 a sin wave에서는 양수값 / c sin wave에서는 음수값 인경우가 많기 때문
b sin wave에서는 양 -> 양 / 음 -> 음 인 부분이 많음 -> 내적 값이 더 높게 나올 수 밖에 없음
어떤 웨이브가 있다고 치면 값이 같으면 내적 값이 높게 나오고 값이 다르면 내적 값이 낮게 나온다. 내적값이 높다 = correlation이 높다

3) 내적값으로 보는 correlation의 오류

시작 점이 다른 wave(주파수<속도>는 same)
하나는 sin wave / 하나는 cos wave (차이 90도 / 1/2π)
-> 둘의 내적 값은 '0'이 된다.
a벡터와 b벡터가 1/2π가 될 때 -> 두 벡터의 내적 값은 0이 된다. (9차원 상의 벡터)
phasor의 이동에 따라 민감하게 반응 한다.
왜냐하면 cos(1/2π)은 0인데 →a・→b=|→a|*cos(θ)*|→b|이 공식을 통해 보자면 내적의 값(분자)가 0일 때 0의 값이 될 수 있기 때문이다.

4) 오류의 해결

민감하게 반응하는 정도를 줄이려면 -> complex phasor을 써서 inner product를 할꺼임
complex한 것과 inner product를 할 것임 -> 이러면 phasor에 대한 민감도를 줄일 수 있음
즉, 2차원이 아닌 3차원과의 결합을 통해 오류를 줄일 것임.