선형대수학 클린업 2주차

[REVIEW]

지난주 우리는 선형대수학의 기본 뼈대를 잡는 시간을 가졌다.
선대의 기본이 되는 **선형성**과, **벡터** 그리고 **행렬**의 개념을 살펴보았으며, 소거법을 이용해 **선형방정식 Ax=b**의 해를 판별하고, 해가 존재하는 경우 그 값을 구하는 법도 공부했다. 선형방정식을 **선형변환**의 차원에서 해석하면서는 행렬에서 교환법칙이 성립하지 않는 이유를 이해하고, 소거법과 역행렬을 기하학적으로 해석하는 법을 익혔다. 마지막으로 선형변환에 이동변환의 개념까지 포함한 **아핀변환**을 다루며, 이것을 바탕으로 딥러닝의 알고리즘을 해석해보았다.

[TODAY's GOAL]

** 기하학적, 공간적으로 이해해봅시다 **

1. **행렬식**
    1) 기하학적 의미
    2) 행렬식과 선형변환
    3) 행렬식의 사용
2. **공간개념 이해하기**
    1) 선형부분공간, Span, Column space
    2) Null space
    3) 독립, 기저
    4) 차원, Rank
3. **투영벡터**
    1) 벡터의 크기(거리)와 직교
    2) 투영벡터
4. **회귀에 적용해보자! : 선형회귀, 가중회귀**

۞ **오늘도 파이팅!** ۞

[Contents]

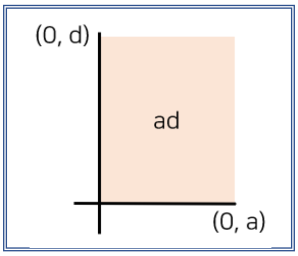
1. 행렬식 (Determinant)

행렬식은 n x n의 **정사각행렬**에 스칼라를 대응시키는 일종의 함수이며, det(A) 혹은 det A라고 표현한다. R의 det. 파이썬의 np.linalg.det 함수를 이용하면 쉽게 계산할 수 있으며, 본 스터디에서는 행렬식의 계산방법은 다루지 않는다. 대신, 행렬식을 기하학적으로 이해하고, 그 쓰임을 살펴보자.

1) 기하학적 의미

행렬식은 기하학적으로 행렬의 역벡터로 만드는 공간의 크기를 의미한다. 구체적으로 살펴보자

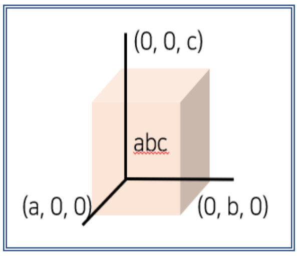
- ✓ 2x2 행렬의 열벡터로 만드는 평행사변형의 넓이



행렬 A가 $\begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix}$ 라고 하자. A의 열 벡터로 가로의 길이가 |a|, 세로의 길이는 |d|인 평행사변형(직사각형)이 만들어지며, 이때의 넓이는 |ad|가 된다. 오른쪽 그림은 a,d가 양수일때의 그림이다.

이때, det(A) 역시 ad인점을 고려하면, **행렬의 열벡터가 만드는 평행사변형의 넓이가 곧 |det(A)|** 임을 이해할 수 있다.

- ✓ 3x3 행렬의 열벡터로 만드는 도형의 부피



행렬 A가 $\begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix}$라고 하자. 행렬 A의 열벡터가 만드는 도형(직육면체의) 부피는 abc이고, 이는 A의 행렬식의 절댓값, |det(A)| 와 같다.

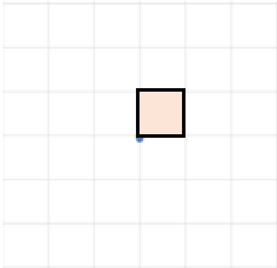
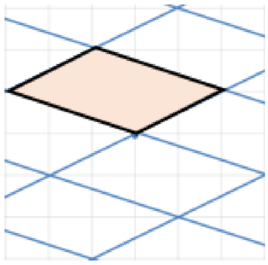
2) 행렬식과 선형변환

이제 행렬식으로 선형변환을 해석해보자.

지난주에 Ax=b를 '벡터x에 행렬A를 곱해 벡터b로 만드는' 선형변환의 관점에서 이해했었다. 이 때 행렬식의 개념을 적용하면, det(A)는 선형변환 전 후의 넓이 관계를 설명해준다. 벡터x로 만드는 공간의 넓이는 선형변환 후 |det(A)|배 변화해 벡터b가 만드는 공간의 넓이가 된다는 것인데, 예시를 통해 이해하자!

$$\begin{bmatrix} 2 & -3 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$$

지난주, 위의 식을 '벡터x에 행렬A를 곱하는 선형변환을 통해 벡터b가 만들어진 것'으로 해석했으며, 기저벡터 (1,0), (0,1)의 조합에서 기저벡터(2,1), (-3,1)의 조합으로 변형, 새로운 공간이 형성되었다고 했다. 그러면 이때 공간은 얼마나 확장 혹은 축소되었을까? 여기서 행렬식의 개념이 들어간다.

선형변환 전 벡터x로는 (1,0), (0,1)인 평행사변형을 만들 수 있으며, 이때의 넓이는 1이다. 한편, 선형변환 후의 벡터 b로는 (2,1), (-3,1)인 평행사변형이 만들어지며 이때의 넓이는 5이다. 선형변환을 통해 넓이가 5배 되었다는 뜻인데, 이때 5는 |det(A)|와 일치한다.

이를 정리하면!

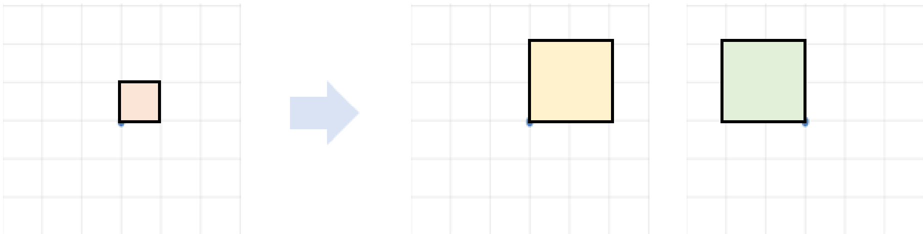
Ax=b에서

(2차원) 변환 후 넓이(b) = |det(A)| x 변환 전 넓이 (x)

(3차원) 변환 후 부피(b) = |det(A)| x 변환 전 부피 (x)

왜 det(A)에 절댓값을 해주나요?

행렬식의 부호는 **공간의 반전 유무**를 의미해요. 아래의 그림을 봐주세요.



노란색으로 변하는 선형변환의 경우는 기저벡터 (1,0), (0,1)을 모두 +2배해여 이때의 A는 $\begin{bmatrix}2 & 0\\0 & 2\end{bmatrix}$라고 할 수 있겠네요. det(A)는 4! 선형변환 이후 넓이는 1에서 4로 4배가 되었어요~
한편, 초록색으로 변하는 선형변환은 기저벡터 (1,0), (0,1)을 각각 -2, +2배 확장시킨 것으로 이때의 A를 $\begin{bmatrix}-2 & 0\\0 & 2\end{bmatrix}$ 이라 하면, det(A)는 -4예요! 평행사변형의 넓이는 |-4|배로, 이전의 선형변환과 면적 변화는 같지만, 좌우반전이 되었다는 점~

그렇다면 det(A)가 0인 것은 무엇을 의미하나요?

det(A) = 0이라는 것은 행렬 공간의 부피가 없음, 즉 **공간을 압축하는 선형변환**임을 의미해요. 3차원 공간에서 평면의 부피는 0이고, 2차원 공간에서 직선의 넓이는 0이라는 해설을 덧붙이면 이해가 좀 더 쉬울까요??

3) 행렬식의 쓰임

A. 역행렬 존재 유무 판별

역행렬이 존재한다는 것은 'Ax=b가 유일한 해를 갖는다(unique)', 'x와 Ax가 서로 일대일 대응이다'와 같은 의미이며, 공간이 압축되는 선형변환의 경우 역행렬이 존재하지 않는다고 했다. 그리고 방금 det(A)=0이 공간압축 선형변환임을 설명했다. 이를 종합하면, **det(A)=0인 경우는 역행렬이 존재하지 않는다.**

B. Cramer's Rule로 해구하기

det(A)가 0이 아닐 때, 즉 A의 역행렬이 존재할 때, Cramer's Rule을 이용하면 간단하게 Ax=b의 x를 구할 수 있다. 예시는 아래와 같으며, 구체적인 설명은 생략한다.

$$\begin{cases} 3x+2y-z=1 \\ x-y+5z=-2 \\ 2x+y=3 \end{cases} \quad \text{is a Cramer's system}$$

$$x=\frac{\begin{vmatrix} 1 & 2 & -1 \\ -2 & -1 & 5 \\ 3 & 1 & 0 \end{vmatrix}}{\det A}=\frac{24}{2}=12; \quad y=\frac{\begin{vmatrix} 3 & 1 & -1 \\ 1 & -2 & 5 \\ 2 & 3 & 0 \end{vmatrix}}{\det A}=\frac{-42}{2}=-21$$

$$z=\frac{\begin{vmatrix} 3 & 2 & 1 \\ 1 & -1 & -2 \\ 2 & 1 & 3 \end{vmatrix}}{\det A}=\frac{-14}{2}=-7 \qquad \begin{cases} x=12 \\ y=-21 \\ z=-7 \end{cases}$$

2. 공간 개념 이해하기

1) 선형부분공간

A. 벡터공간

여러 벡터들이 모여 어떠한 공간을 형성할 때, 그것을 벡터공간이라고 한다. $\mathbb{R}^n$이 그 대표적인 예이다.

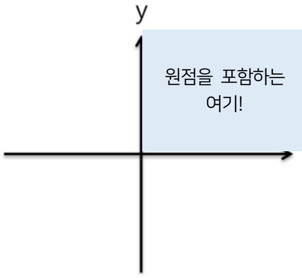
B. 선형부분공간 (subspace)

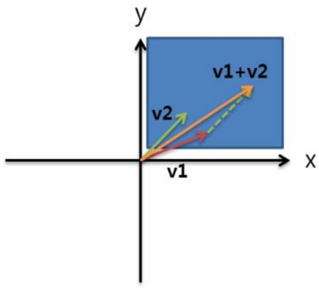
벡터공간 안의 몇 가지 벡터의 집합으로 이루는 벡터공간 안의 또 다른 부분 벡터공간을 선형부분공간이라고 한다. '선형'에 의해서 선형부분공간은 아래와 같은 조건을 만족해야 한다.

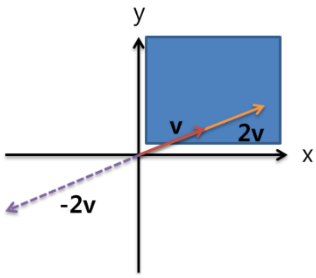
1. 0벡터(원점)가 존재한다.
2. 선형부분공간에 있는 벡터에 다른 벡터를 **더했을 때**, 그 합도 선형부분공간에 존재한다.
3. 선형부분공간에 있는 벡터에 **스칼라를 곱했을 때**, 그 곱도 선형부분공간에 존재한다.

원점조건은 갑자기 어디서 나타난거죠?
어떤 벡터에 0을 곱하면 0이 되기 때문에, 선형부분공간은 반드시 원점을 포함하고 있어야 해요!

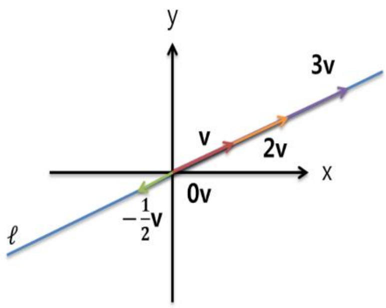
예시를 통해 선형부분공간을 이해해보자. 2차원 공간에서 축을 포함한 1사분면은 선형부분공간이 될 수 있을까? 축을 포함한 1사분면은 원점을 포함하고, 1사분면 내의 벡터끼리 더하면 1사분면에 위치해 두번째 조건까지 만족한다. 하지만, 1사분면 내의 벡터에 음수 스칼라를 곱하면 그 벡터는 1사분면을 벗어나 세번째 조건이 충족되지 않는다. 결국, 축을 포함한 1사분면은 선형부분공간이라고 할 수 없다.







한편 오른쪽 그림의 부분공간 l은 원점을 포함하면서 v에 대해 덧셈조건과 상수배 조건까지 만족하므로, **직선 l은 2차원 공간의 선형부분공간**이다.

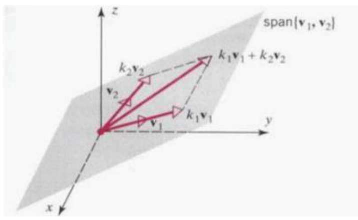


C. Span

Span은 **a1, a2, ⋯ , an의 모든 선형결합을 조합한 공간**을 의미한다. 이제는 이것을 벡터 a1, a2, ⋯ an의 모든 선형결합이 속하는 **선형부분공간**이라고 이해할 수도 있다.

앞에서는 2차원 공간에 있는 직선 l을 벡터v의 상수배와 합, 그리고 원점을 포함하고 있는지를 중심으로 살펴보고 이를 선형부분공간이라 정리했다. 하지만, 직선l은 '모든 v에 대한 조합'이라는 점에서 span{v}라고도 해석할 수 있다. 결국 직선l 은 2차원 공간의 선형부분공간임과 동시에 span{v}인 것이다.

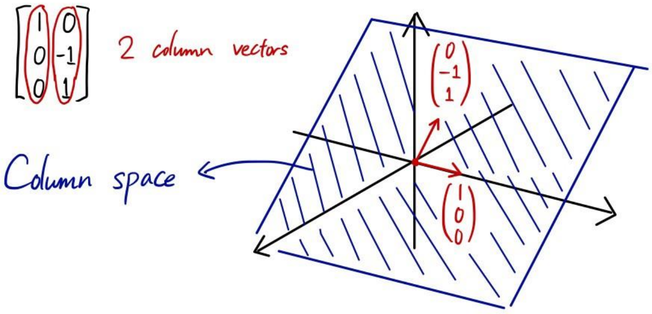
u와 v가 3차원 공간의 벡터일 때도 이해해보자. v1과 v2의 스칼라배와 합으로 만든 모든 조합은 span{v1,v2}가 되며, 이것은 3차원 공간에서 원점을 지나는 평면으로 존재한다.



사실, span은 사용하는 벡터에 따라서 모든 공간을 채울 수도 있고, 위에서 본 예시처럼 2차원 공간의 직선 혹은 3차원 공간의 평면인 선형부분공간일 수도 있다. 따라서 span을 특정 공간의 특정 형태로 기억하기 보다는 '**벡터들의 가능한 선형결합의 모음을 한 공간에 몰아넣은 것**'이라는 정의적 차원으로 이해해주었으면 한다.

D. Column space

A의 열벡터의 연산으로 만드는 공간(선형부분공간), 즉 **A열벡터의 span**을 column space라 한다.

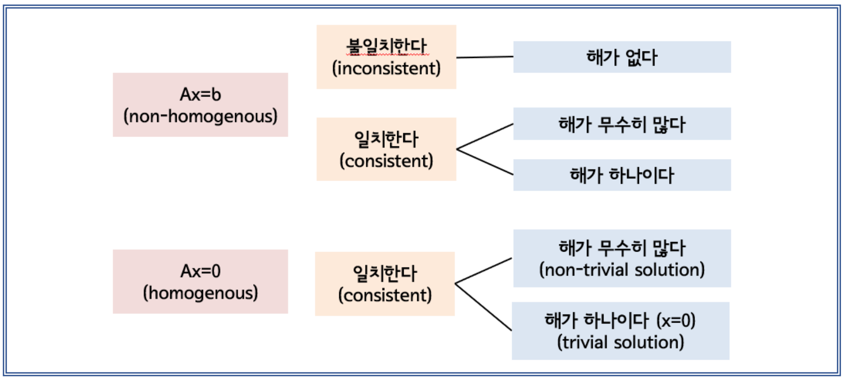


Col(A)는 A에 대한 모든 선형결합, 즉 모든 Ax를 포함한다. 때문에 Ax=b가 있다는 것은 벡터b가 col(A)에 있다는 것을 의미하며, 이를 바탕으로 b가 span{a}에 있어야 함도 설명할 수 있다. Ax=b가 모든 b에 대해 해를 갖는다는 것은, Col(A)이 모든 b를 표현함을 뜻하며 결국 col(A)는 $\mathbb{R}^n$이 된다.

2) Null Space

A. Ax=0

지난주에 다뤘던 해의 종류에 살을 조금 붙여보자. Ax=b 에서 b 가 0 인 경우, 즉 Ax=0 인 경우를 homogeneous 라 하고, 그렇지 않을 때를 non-homogeneous 라고 한다. Homogeneous 인 방정식은 항상 해가 존재하며, x 가 모두 0 인 유일한 해를 갖는 경우를 특히 trivial 이라 한다.



B. Null space

Ax=0를 만족하는 해 x가 이루는 공간, 즉 homogeneous한 방정식의 해의 집합을 Null Space(영공간)이라고 한다. 선형변환의 관점에서는 이것을 A와 곱한 결과가 0 되는 벡터x의 집합으로 해석한다

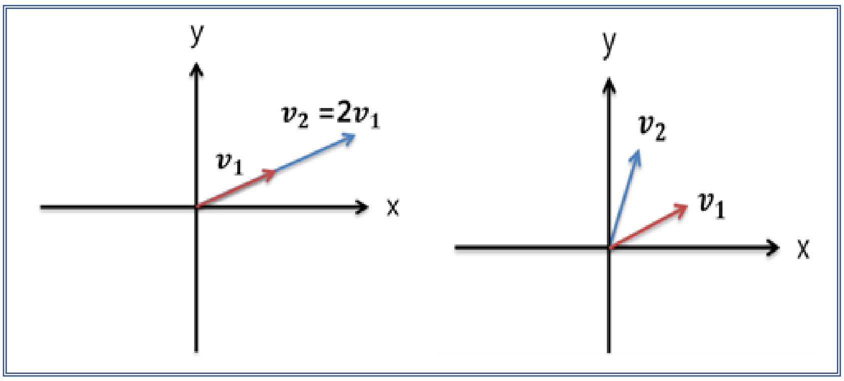
$$\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 2 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

위와 같은 방정식 Ax=0에서 식을 만족시키는 해 (x,y,z)는 (0,0,0), (1,1,-1) (-2,-2,2) 등이 여러 조합이 있는데, 이들을 모아 형성한 공간이 바로 Null space이다. 이때 해에 0벡터가 포함되고, 해벡터의 상수배나 해벡터끼리의 합도 해의 집합에 포함되기 때문에 Null space도 **선형부분공간** 중 하나가 된다. 결국 방정식의 해벡터가 $\mathbb{R}^n$ 이라면, 그것의 null space는 $\mathbb{R}^n$의 선형부분공간이 된다.

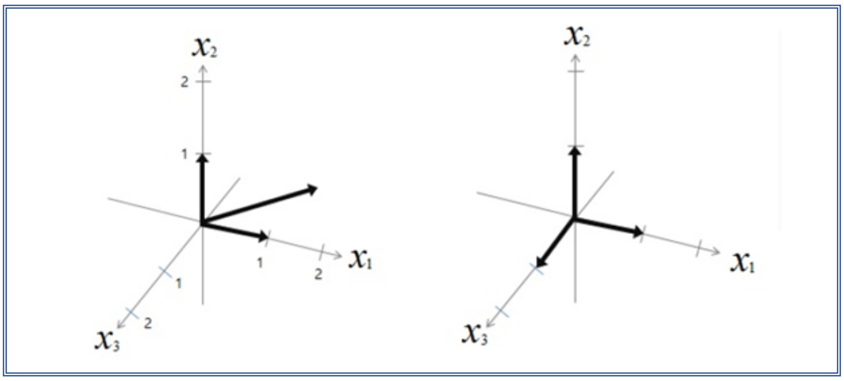
3) 기저

A. 선형독립(independence)

$c_1x_1 + c_2x_2 + \dots + c_nx_n = 0$에서 **c가 모두 0인 경우를 제외한 어떠한 ci의 조합으로도 식이 성립되지 않는 경우**, 벡터 x들을 독립이라고 한다. 방정식차원으로 보면, **Ax=0을 만족하는 벡터x가 오직 0벡터뿐 일** 때(trivial solution만 가질 때), A의 각 열벡터들을 선형독립이라고 한다. 예시를 통해 이해해보자.



왼쪽부터 살펴보자. 왼쪽 그림과 같이 벡터 v1,v2가 있을 때, 이들은 서로 독립일까?
c1v1+ c2v2 = 0이라는 선형결합은 c1=2, c2=-1 일 때 성립한다. 0벡터가 아닌 벡터c(2,-1)에 의해 식이 성립되므로 벡터v1과 v2는 독립이 아니다. 종속관계이다.
한편, 오른쪽 그림에 경우, 두 벡터 v1과 v2를 이용하여 어떠한 선형결합을 해도, c가 모두 0인 경우를 제외하고는 0을 만들 수 없다. 때문이 이 경우 v1과 v2는 독립이다.



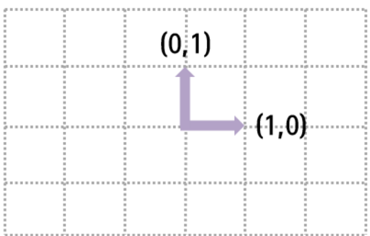
3차원 공간에서도, 왼쪽의 (1,0,0) (0,1,0). (2,1,0)처럼 (0,0,0)이외에 해 (여기서는 (2,1,-1))가 존재하고, 하나의 벡터를 다른 벡터의 조합으로 표현 할 수 있으면 선형종속이다. 오른쪽 그림의 벡터는 (1,0,0), (0,1,0), (0,0,1)로, 이들은 0벡터를 유일한 해로 갖는, 선형독립관계이다.

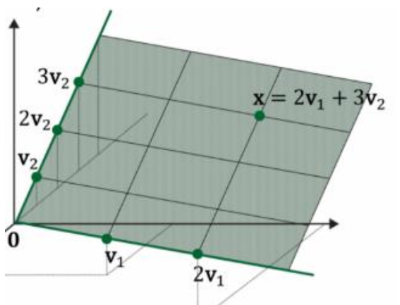
선형독립을 **Null space에 0벡터만 있을 때**로 해석하기도 하는데, 이는 결국 Homogeneous한 방정식의 해의 조합으로 만드는 공간에 오직 0벡터만 있다는 것을 뜻하며 앞의 정의와 같은 맥락이다.

B. Basis (기저)

기저는 앞서 다룬 span과 선형독립을 모두 포함하는 개념으로, 어떤 공간을 구성하는 span벡터의 최소 집합을 기저라고 한다. 특정 공간을 구성하는 데에 이미 존재하는 벡터들의 조합으로 설명되는(종속) 벡터의 추가는 의미가 없기 때문에 '최소 집합'이라는 개념이 등장하는데, 이것이 곧 어떠한 벡터를 선택해 조합하더라도 다른 벡터를 만들 수 없는 선형독립과 닿아 있는 것이다. 우리는 **벡터공간을 효율적(최소)으로 표현할 수 있는 벡터들(span)**이라고 기저를 이해하자**.**

지난 클린업에서 언급한 (1,0), (0,1)과 같은 조합이 기저의 대표적인 예이며, 이러한 조합을 표준기저 (standard basis)라고 한다. 하지면 여기서 중요한 것은 기저에 표준기저만 존재하는 건 아니라는 사실! **어떤 공간을 span하는 기저는 유일하지 않다.** 아래 그림은 각각 표준기저와 표준기저가 아닌 기저의 예이다.





C. Null space와 Column space의 기저

✓ Column space의 기저

Column space의 기저는 피벗이 없는 열이다.

$$A = \begin{bmatrix} 1 & 1 & 0 & 2 \\ -2 & -2 & 1 & -5 \\ 1 & 1 & -1 & 3 \\ 4 & 4 & -1 & 9 \end{bmatrix} \quad \sim \text{RREF} \quad \begin{bmatrix} 1 & 1 & 0 & 2 & : & 0 \\ 0 & 0 & 1 & -1 & : & 0 \\ 0 & 0 & 0 & 0 & : & 0 \\ 0 & 0 & 0 & 0 & : & 0 \end{bmatrix}$$

이때 자유변수가 없는(pivot이 있는) (1,0,0,0), (0,1,0,0)이 col(A)의 기저이다.

Column space의 기저의 개수 = pivot(leading1)의 개수

✓ Null space의 기저

Null space의 기저는 **자유변수를 상수로 하는 선형조합의 벡터**들로, 기저를 통해 해를 효율적으로 구성할 수 있다.

위의 예시와 같은 행렬 A에 대해서 RREF로 정리 후 구한 해와 이때의 기저(S)는 아래와 같다.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -x_2 - 2x_4 \\ x_2 \\ x_4 \\ x_4 \end{bmatrix} = \begin{bmatrix} -s-2t \\ s \\ t \\ t \end{bmatrix} = s\begin{bmatrix} -1 \\ 1 \\ 0 \\ 0 \end{bmatrix} + t\begin{bmatrix} -2 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \ (s,\ t \in \mathbb{R}) \qquad S = \left\{ \begin{bmatrix} -1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -2 \\ 0 \\ 1 \\ 1 \end{bmatrix} \right\}$$

Null space의 기저의 개수 = 자유변수 개수 = n-(pivot의 개수)

	Column space	Null space
dimension	pivot 의 개수	pivot이 없는 열의 개수 = 자유변수의 개수

4) 차원과 Rank

차원은 벡터공간에서 기저 벡터의 개수를 기준으로 정한다. Column space의 차원은 Rank라 한다.

$$A = \begin{bmatrix} 2 & 5 & -3 & -4 & 8 \\ 4 & 7 & -4 & -3 & 9 \\ 6 & 9 & -5 & 2 & 4 \\ 0 & -9 & 6 & 5 & -6 \end{bmatrix}$$

SOLUTION Reduce A to echelon form:

$$A \sim \begin{bmatrix} 2 & 5 & -3 & -4 & 8 \\ 0 & -3 & 2 & 5 & -7 \\ 0 & -6 & 4 & 14 & -20 \\ 0 & -9 & 6 & 5 & -6 \end{bmatrix} \sim \cdots \sim \begin{bmatrix} 2 & 5 & -3 & -4 & 8 \\ 0 & -3 & 2 & 5 & -7 \\ 0 & 0 & 0 & 4 & -6 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Pivot columns

The matrix A has 3 pivot columns, so rank $A = 3$.

3. 투영벡터

1) 벡터의 크기와 거리

$\mathbb{R}^n$의 벡터 x = (x_1, x_2, …, x_n)에 대하여 벡터의 크기(norm, length, ||x||)는 $\sqrt{{x_1}^2+{x_2}^2+\ldots+{x_n}^2}$로 구한다. 벡터의 크기는 점X($x_1$, x_2, …, x_n)와 원점 사이의 거리로 정의할 수 있는데, 이것을 점 X와 Y사이의 거리로 일반화하면 거리 ||x - y|| = $\sqrt{(x_1-y_1)^2+(x_2\ -\ y_1)^2+\ldots+(x_n-y_n)^2}$가 된다.

2) 내적 (dot product)

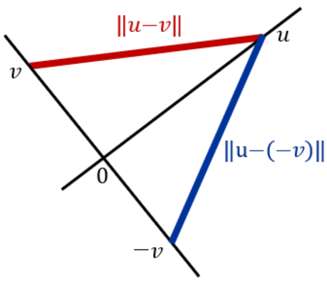
$\mathbb{R}^n$의 벡터 x = (x1, x2, …, xn), y = (y1, y2, ⋯, yn)에 대하여, 실수 x1y1+x2y2+ ⋯ + xnyn 와 같이 계산하는 것을 내적이라고 한다. 원소의 개수가 같은 x, y에 대해, 벡터x를 transpose한 뒤 y를 곱하면 1x1의 상수가 나온다.

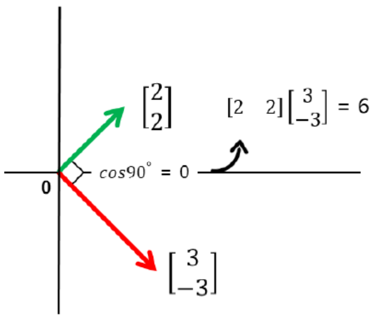
$$\mathbf{u}=\begin{bmatrix}3\\2\\1\end{bmatrix},\mathbf{v}=\begin{bmatrix}1\\3\\5\end{bmatrix},\mathbf{u}\cdot\mathbf{v}=\underbrace{\mathbf{u}^T\mathbf{v}}_{\text{inner product or dot product}}=\begin{bmatrix}3&2&1\end{bmatrix}\begin{bmatrix}1\\3\\5\end{bmatrix}=[14]$$

$$(1\times 3)(3\times 1)=1\times 1$$

3) 직교 (Orthogonality)

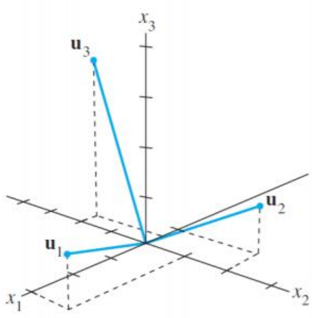
벡터가 서로 수직으로 만날 때를 직교라고 하며, 기하학적으로 직교는 두 벡터 **u와v의 거리와 u와 -v의 거리가 같음**을 뜻한다. 수식상으로는 **u와 v를 내적해서 그 값이 0인 경우**를 직교라고 한다.



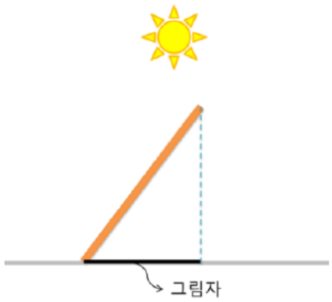


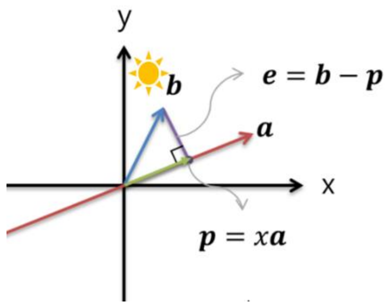
서로 직교하는 벡터들의 집합을 orthogonal set, 직교집합이라 한다. orthogonal set의 벡터들은 서로 선형 독립이며, 이들의 span으로 만들어지는 벡터공간의 basis가 된다. 지금까지의 내용과 비슷한 맥락이니 아래 예시를 보고 공간적 그리고 직관적으로 이해해보자.

$$\mathbf{u}_1=\begin{bmatrix}3\\1\\1\end{bmatrix},\quad \mathbf{u}_2=\begin{bmatrix}-1\\2\\1\end{bmatrix},\quad \mathbf{u}_3=\begin{bmatrix}-1/2\\-2\\7/2\end{bmatrix}$$



4) 투영벡터 (projection)

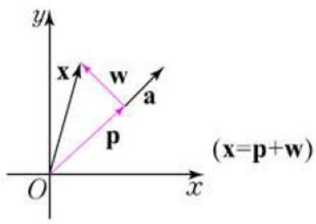




왼쪽 그림과 같은 상황에서, 주황색 막대는 태양과 수직인 땅에 그림자를 갖는다. 태양의 관점에서는 길이가 같은, 땅 위의 막대를 찾았다고도 할 수 있겠다. 이 개념을 벡터, 공간차원에서 이해하면 그것이 투영벡터이다.

투영이란 하나의 벡터를 다른 벡터로 옮겨서 표현하는 것을 말한다. 오른쪽 그림의 경우 벡터 b가 막대, a가 땅, 그리고 p가 그림자에 대응하며, 이때의 그림자 벡터p를 투영벡터라 한다. 벡터b를 직선a 위의 벡터로 매핑했다는 것은 벡터b의 직선 공간으로의 공간압축으로 해석할 수 있으며, 앞서 반복적으로 언급했던 '**공간을 압축하는 선형변환**'이 바로 이 경우이다. 한편 태양과 땅이 수직이었다는 사실을 고려하면, p는 a와 b사이의 **최소거리**에 위치함도 알 수 있다.

오른쪽 그림과 같을 때, 벡터p를 '벡터x의 span{a}상의 정사영'이라고 하며, $\text{Proj}_a x$ 라는 기호로 나타낸다. 그 값은 실제값(x)와 예측값(p)의 차이인 w (x-p)와 벡터a가 직교함을 이용하여 계산하지만, 자세한 풀이과정은 본 스터디에서 다루지 않겠다.



$$p = ta = \frac{x \cdot a}{\| a \|^2} a = \text{proj}_a x$$

$$w = x - p$$

4. 회귀에 적용해보자! : 선형회귀, 가중회귀

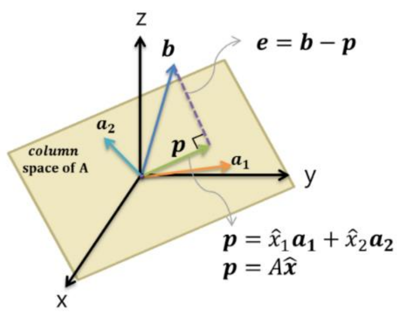
1) 투영벡터와 Least Square Method

지난주, 선형대수학에서는 연립선형방정식의 해를 찾는 것을 연구의 대상으로 삼는다고 했다. 그와 함께 연립선형방정식의 해의 종류는 크게 세 가지뿐이며, 해가 있는 경우(consistent)는 그 해를 구하는 과정까지 살펴봤다. 그렇다면 미지수보다 방정식이 더 많은 경우, 즉 Ax=b의 해가 없는 경우는 무엇을 연구 대상으로 삼아야할까? 이 경우 주어진 방정식을 최대한 만족하는 일반화되는 해를 찾아서 사용하는데, 이때 방금 살펴본 거리개념과 투영벡터가 등장한다.

왜 투영벡터를 이용해야 하나요?

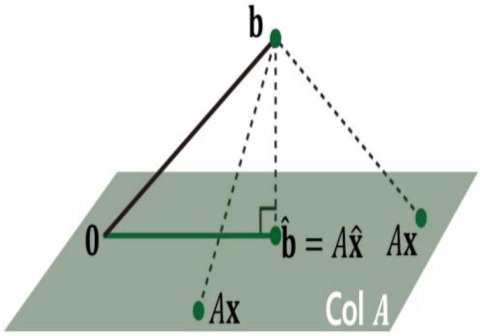
Ax=b의 해가 존재하려면 b가 span{a}에 있어야 한다는 개념이 바로 여기 쓰여요!

오른쪽 그림은 a1과 a1로 만든 col(A)에 b가 위치하지 않는, Ax=b의 해가 없는 상황이죠~ b는 연구하고 싶고, 근데 b는 col(A)에 없으니 최대한 원래식과 비슷한 대역을 찾았는데, 그게 최소거리 투영 벡터인 것이랍니다!



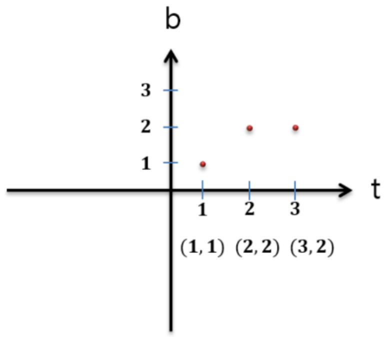
왜 제일 비슷한 벡터를 찾는데 최소거리, 근접해를 찾는거죠?

그 답은 수직에 있어요! 해가 있는 경우, Ax와 b의 거리(e)는 0이기 때문에 우리는 가능한 b에 가까운, b인 척하는 벡터를 찾기 위해서 b와 col(A)의 최소거리에 주목하는 거예요! 아래의 그림과 같은 상황이라면 b에서 수직으로 내린 경우가 가장 그 길이가 짧겠죠? 그래서 수직으로 내린 벡터, 투영벡터를 사용하는 겁니다!



- ✓ 최소자승법(Least Square Method)

Least Square Method란 데이터를 최대한 만족하는 하나의 line에 대한 식을 찾는 방법이다. 이렇게 찾은 line은 데이터의 경향과 예측이 가능케한다. 예시로 이해해보자



$$\begin{matrix}\text{"ax+b=y"}\\ 1a+b=1\\ 2a+b=2\\ 3a+b=2\end{matrix} = a\begin{pmatrix}1\\2\\3\end{pmatrix}+b\begin{pmatrix}1\\1\\1\end{pmatrix}=\begin{pmatrix}1\\2\\2\end{pmatrix} = \begin{pmatrix}1&1\\1&2\\1&3\end{pmatrix}\begin{pmatrix}a\\b\end{pmatrix}=\begin{pmatrix}1\\2\\2\end{pmatrix} \Rightarrow \begin{pmatrix}1&1\\1&2\\1&3\end{pmatrix}\begin{pmatrix}\hat{a}\\\hat{b}\end{pmatrix}=\begin{pmatrix}\widehat{y1}\\\widehat{y2}\\\widehat{y3}\end{pmatrix}$$

데이터를 Ax=y 형태로 정리한 뒤, 해를 구할 수 있는 $A\hat{x}=\hat{y}$으로 바꾸는 과정

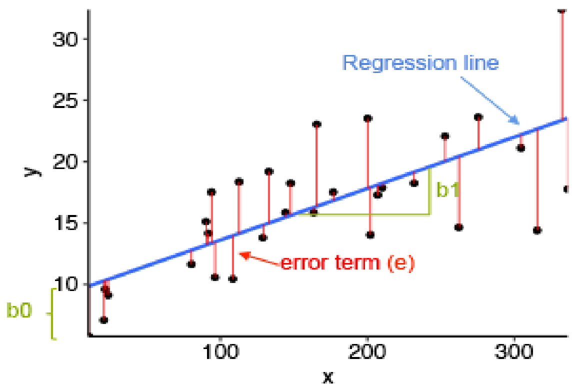
(1,2), (2,2), (3,2)인 데이터가 있을 때 이것을 그래프로 나타내면 위의 그림과 같고, 선형방정식 Ax=b 꼴로 정리하면 아래의 과정을 거친다. 이 선형방정식은 문자보다 방정식이 많아 해가 존재하지 않기 때문에 우리는 이에 근사한 해(best solution)을 구해야 한다. **Least Square Method** 에서는 이것을 각 **데이터와 직선 사이에 에러의 제곱합이 최소가 되게끔 하는 직선을 찾는 것**으로 한다.

Ax위에 b가 존재하지 않아 해를 구할 수 없을 때,
Ax 와 b의 최소거리를 바탕으로 간접해 $\hat{x}$을 구하는 것을 Least Square Method이라 하고,
이를 통해 b의 투영벡터인 $A\hat{x}$을 이용해 간접해를 찾는다

2) 선형회귀분석

회귀분석은 통계에서 다루는 가장 기본적인 모델로, 설명변수 x와 종속변수y간의 관계를 파악하고, test set의 y를 예측하는데 쓰인다. 지난주 패키지 모델링 문제를 떠올리면 이해가 쉬울 것이다. 앞서 다룬 LSM을 주어진 데이터를 이용해 회귀를 한 것이라고 이해할 수도 있겠다. 이보다 더 자세한 내용은 회귀분석 클린업 참고

회귀식의 예측력이 높고, x와 y관계를 잘 설명하기 위해서는 실제y와 예측한 $\hat{y}$이 비슷해야 하는데, 이때 비슷하다를 거리가 가깝다로 보고 **y-$\hat{y}$(residual)의 제곱값을 최소화하는 회귀식**을 만든다.



✓ 잔차제곱을 최소화하는 베타계수 찾기

(1) 회귀식 $y = \beta 0 + \beta 1x$을 Ax=b의 형태, 즉 $X\beta = y$로 바꾼다.

Predicted y-value		Observed y-value
$\beta_0 + \beta_1 x_1$	=	y_1
$\beta_0 + \beta_1 x_2$	=	y_2
$\vdots$		$\vdots$
$\beta_0 + \beta_1 x_n$	=	y_n

We can write this system as

$$X\beta = \mathbf{y}, \quad \text{where } X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}, \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

(2) 모든 예측값과 실제값이 일치하지 않기 때문에 $X\beta$=y은 해가 존재하지 않으므로, Least-Square method를 적용해 $X\beta$와 y의 최소거리인 간접해를 구한다.

$$\begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix}=\begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{pmatrix}\begin{pmatrix} \hat{a} \\ \hat{b} \end{pmatrix}=\begin{pmatrix} \widehat{y1} \\ \widehat{y2} \\ \widehat{y3} \end{pmatrix}$$

해를 구할 수 없는 Ax=b에서 해를 찾을 수 있는 방정식 A$\hat{x}$=$\hat{y}$으로 대체한다.

Ax = b 일 때 x 의 해

$$A^T A\mathbf{x} = A^T b$$

$X\beta = y$ 일 때 β의 해

$$X^T X\beta = X^T y$$

식계산은 프로그램을 이용하면 쉽게 구할 수 있으니, 계산보다는 반복해서 강조하고 있는 투영벡터와의 관련성에 방점을 두고 이해하도록 하자.

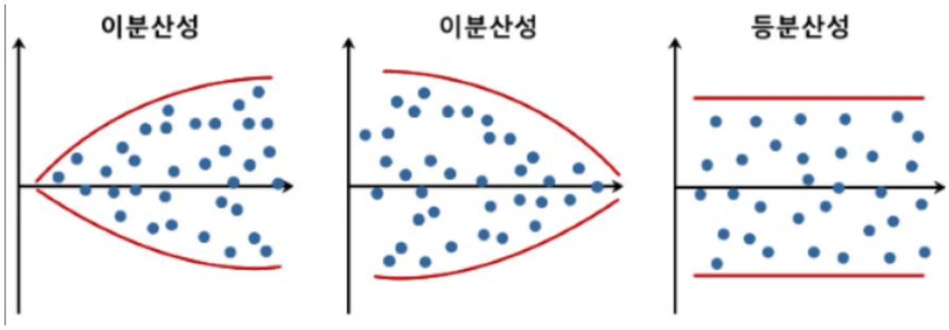
> 회귀분석에서는 Least-Square problem을 통해 회귀식 $X\hat{\beta} = \hat{y}$ 의 간접해 $\hat{y}$(proj y)를 구함으로써, 실제값($\boldsymbol{y}$)과 예측값($\hat{y}$, $X\hat{\beta}$)의 거리(residual)를 최소화하는 회귀식을 구한다.

Ax=y 와 y=Xβ에 모두 x, y 가 들어가서 혼란스러워요
Ax=y : 해(x)가 존재하지 않는다 = y 가 Ax 위에 없다 → A$\hat{x}$=$\hat{y}$; 이때 A 는 선형변환 행렬
Xβ=y : y 가 Xβ위에 없다 → $X\hat{\beta} = \hat{y}$; 이때 x 는 [1 데이터의 독립변수 데이터 모음]
y 예측(투영,근접해)은 공통이고, 위의 x 와 아래의 β(구하고싶은거) / 위의 A 와 아래의 X(주어진거) 가 같은 맥락이다.

3) 가중회귀모델

단순선형회귀식은 yi= $\beta 0 + \beta 1x1 + \epsilon i$ 으로, 이때 ϵi 은 지금까지 말한 residual, 즉 오차항이다. ϵi 는 평균이 0, 분산이 σ^2인 상황, 즉 입력변수와 무관하게 무작위적으로 골고루 분포한 등분산성을 가정하는데. 오차항의 분산이 일정하면 더 정확한 오차 계산과 예측이 가능하기 때문이다.

아래의 그림처럼 등분산성 가정이 만족하지 않을 때는, 작은 분산에 가중치를 두어 등분산성을 맞춰주는 처리를 하게 되는데, 이러한 선형회귀를 가중선형회귀라고 한다.



🌞등분산성가정이 만족하지 않을 때는?

(1) log(Y), 1/Y, sqrt(Y)등의 변수변환을 하거나 (2)가중치행렬로 등분산성 가정을 처리하는데, 실제로 가중치행렬을 이용할 때는 분산의 역수를 사용하는 것이 일반적입니다~ 다만! 분산을 알기 어렵기 때문에 경험적으로 정해야 하는데, 자세한 내용은 회귀 클린업을 참고하세요~

- ✓ 가중회귀 베타계수찾기

일반선형회귀에서 $X\beta=y$라는 선형방정식에 Least-square problem을 적용해 베타계수를 추정(OLS)한 것과 달리, 가중회귀에서는 양변에 각각 가중치를 곱한 뒤 Least-square problem을 적용해 베타계수를 추정한다.

$$\begin{pmatrix} w_1 & 0 & \dots & 0 \\ 0 & w_2 & \dots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \dots & w_n \end{pmatrix}\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}\begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{pmatrix} w_1 & 0 & \dots & 0 \\ 0 & w_2 & \dots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \dots & w_n \end{pmatrix}\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Ax = b 일 때 x의 해

$$A^T A\mathbf{x} = A^T b$$

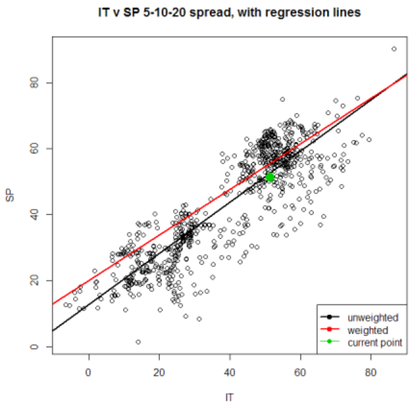
$X\beta = y$ 일 때 β의 해

$$X^T X\beta = X^T y$$

$WX\beta = Wy$ 일 때 β의 해

$$(WX)^T(WX)\beta = (WX)^T Wy$$

✓ 선형회귀, 가중선형회귀 플랏



여기서 current point가 무슨 의미죠?

이 그림은 선형회귀와 가중선형회귀의 차이를 시각적으로 보여주는 플랏입니다! 이때 current point는 실제 IT에 대한 Sp값을 의미하는데요, 플랏 그린이는 이 녹색 점을 통해서 가중선형회귀가 항상(모든 경우에) 선형회귀보다 좋은 성능을 보이는 것은 아니라는 사실을 알려주고 싶었다고 해요!!

✓ 선형회귀, 가중선형회귀 베타계수식, 잔차제곱합 비교

	일반회귀	가중회귀
SSE	$\|y-\hat{y}\|^2$ $=\|y-X\hat{\beta}\|^2$	$\|Wy-W\hat{y}\|^2$ $=\|Wy-WX\hat{\beta}\|^2$
β 의 해	$X^TX\hat{\beta}=X^Ty$	$(WX)^TWX\hat{\beta}=(WX)^TWy$

$X^TX\hat{\beta}=X^Ty$ 에서 t(x)x가 invertible하지 않으면 어떻게 하지?

사실 non-invertible한 경우가 많지 않을 뿐만 아니라, 프로그램에서는 역행렬 존재의 유무와 관계없이 계산을 해줍니다! 하지만! 만약을 위해 설명합니다. X^TX 가 invertible 하지 않은 이유는 크게 두가지로 나뉩니다. (1) 불필요한(중복) feature을 가진 경우, (2) 과도하게 많은 feature을 가진 경우. 각각 살펴봅시다

(1) 불필요한(중복) feature을 가진 경우

x1 : 키, 단위 cm
x2 : 신장, 단위 m
x3 : 키, 단위 feet

위의 예시처럼 같은 의미를 갖는 데이터를 사용하는 경우, 중복된 features 문제가 발생합니다. 이것이 원인인 경우는 중복된 feature를 찾아 제거하면 해결됩니다.

(2) 과도하게 많은 feature 을 가진 경우

주어진 data set 의 크기에 비해 너무 많은 feature 를 사용하는 경우 발생합니다. 예를 들어 주어진 데이터는 10 개인데, 각 데이터의 feature 은 100 개다? 이럴 때는 (a) 일부 features 를 삭제해 줄이는 방법 (b) regularization(정규화) 하는 방법 두가지의 방법이 있습니다.

이때 정규화란 회귀계수에 제약을 가하는 것으로, 과적함(overfitting) 문제를 막기 위해 주로 사용하는 방법입니다. 구체적으로 회귀계수에 어떻게 패널티를 주는지에 따라 Ridge, Rasso, ElasticNet으로 구분하지만, 그것은 회귀 클린업에서 다뤄야하는 일!! 선대에서는 여기까지만 하겠습니다.

🌞 R 의 내장데이터 Carseats 을 이용해 Age(x)로 Sales(y)를 예측하는 회귀모델링을 해보자!

Sales	Age
9.50	42
11.22	65
10.06	59
7.40	55
4.15	38

선형회귀모델

$$X = \begin{pmatrix} 1 & 42 \\ 1 & 65 \\ 1 & 59 \\ 1 & 55 \\ 1 & 38 \end{pmatrix} \quad \beta = \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} \quad y = \begin{pmatrix} 9.50 \\ 11.22 \\ 10.06 \\ 7.40 \\ 4.15 \end{pmatrix}$$

$$X^T X\hat{\beta} = X^T y$$

$$\therefore \hat{\beta} = \begin{pmatrix} \widehat{\beta_0} \\ \widehat{\beta_1} \end{pmatrix} = \begin{pmatrix} -0.83 \\ 0.18 \end{pmatrix}$$

가중선형회귀모델'

나이가 50이상인 데이터에 2배 가중치를 둘까?

$$X = \begin{pmatrix} 1 & 42 \\ 1 & 65 \\ 1 & 59 \\ 1 & 55 \\ 1 & 38 \end{pmatrix} \quad \beta = \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} \quad y = \begin{pmatrix} 9.50 \\ 11.22 \\ 10.06 \\ 7.40 \\ 4.15 \end{pmatrix} \quad W = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

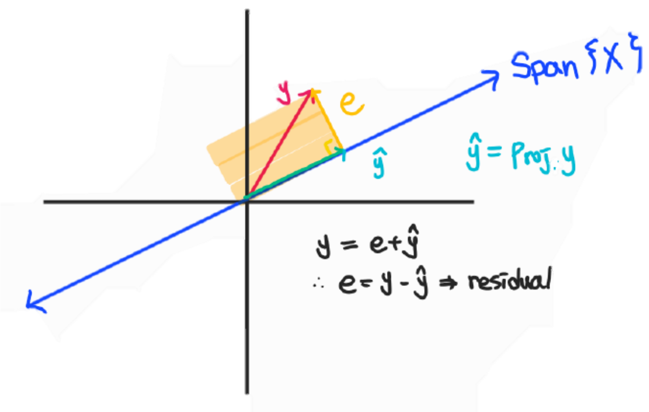
$$(WX)^T WX\hat{\beta} = (WX)^T Wy$$

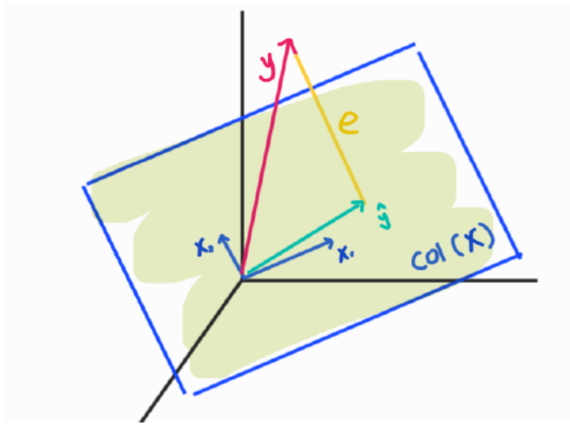
$$WX = \begin{pmatrix} 1 & 42 \\ 2 & 130 \\ 2 & 118 \\ 2 & 110 \\ 1 & 38 \end{pmatrix} \quad Wy = \begin{pmatrix} 9.50 \\ 22.44 \\ 20.12 \\ 14.80 \\ 4.15 \end{pmatrix} \quad \therefore \hat{\beta} = \begin{pmatrix} \widehat{\beta_0} \\ \widehat{\beta_1} \end{pmatrix} = \begin{pmatrix} -2.37 \\ 0.20 \end{pmatrix}$$

♥ 회귀와 투영벡터 개념 다시 정리할게요!

투영벡터 : $proj_x y$ = 벡터 y 를 span{x}에 정사영함 = span{x} 위에 벡터 y 와의 거리가 최소인 간접해 $\hat{y}$ 을 찾음

= 오차e (y-$\hat{y}$)와 span{x}는 직교 = e• x가 0



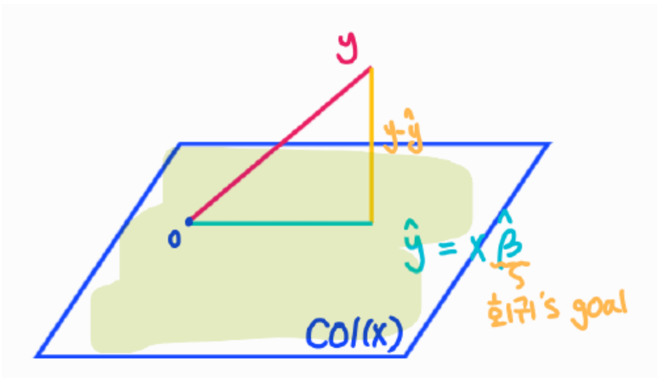


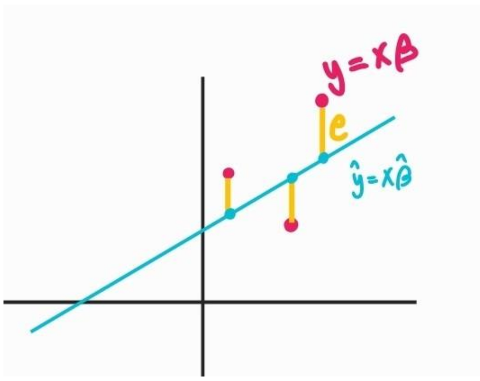
xβ=y 의 해가 없다 ⇔ y 가 span{x}위에 없다. ⇔ y 가 col(x)위에 없다

LSM : Xβ=y 해가 없는 경우 xβ 와 y 의 최소거리를 바탕으로 간접해 $\hat{y}$을 구하는 것

선형회귀분석 : Xβ=y -- Least-Square problem -- > $X\hat{\beta} = \hat{y}$

-- > 간접해 $\hat{\beta}$ 찾기 = 예측값과 실제값의 거리(residual)를 최소화하는 회귀선의 회귀계수 찾기!





회귀분석은 y 를 projection ($\hat{y}$)시켜 최종적으로는 최적(잔차제곱의 최소)의 베타계수를 찾는 것!!

꿩 대신 닭 ⇔ proj ⇔ regression

[Code]

행렬식 계산하기

1~16을 원소로 하는 행렬의 행렬식을 구해봅시다.

파이썬으로 선형회귀식의 계수 구하기

앞에서 살펴본 (1,1), (2,2), (3,2)인 데이터의 선형회귀식을 구해봅시다.

R로 선형회귀식 / 가중선형회귀식의 계수 구하기

앞에서 살펴본 내장데이터 Carseats에 대한 예시를 코드로 구현해봅시다

[다음주 예고]

차원과 차원축소

고유값 고유벡터

SVD

PCA와 요인분석

AHP

** 응용시간 가져봅시다**

끝! 고생하셨습니다!