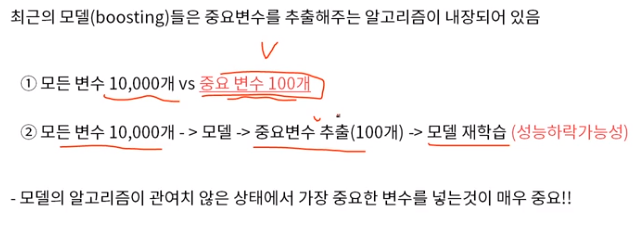
**<feature selection>**

Y의 변동성을 설명하는 유의미한 x변수들을 구하는 것이 모델보다 중요하다

다중공선성을 줄여야한다.

CNN의 해주는 역할은 이미지의 feature를 잘 뽑기 위한 것뿐이다. 이 값을 통해서 학습을 시킨다. NN과의 차이점은 feature(input)이 다르다. 이미지 분류문제에서는 CNN이 항상 NN보다 좋다. Graphical한 feature를 뽑았기 때문에 성능이 좋다. -> 같은 알고리즘이어도 feature가 중요하면 성능이 잘나온다.

?어차피 머신러닝 알고리즘이 중요변수를 알아서 뽑아주는데 왜 이걸 알아야하나?



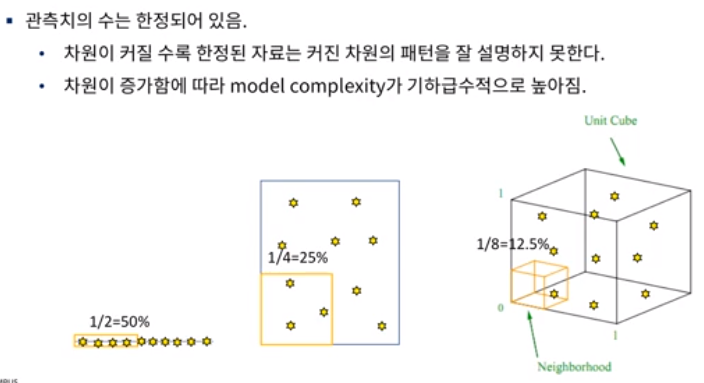
변수선택 방법 : 전진선택법, 후진제거법, stepwise

계수축소법 : ridge, lasso, elastic-net -> SSE를 최소화하는 방향으로

**<PCA – 차원축소>**

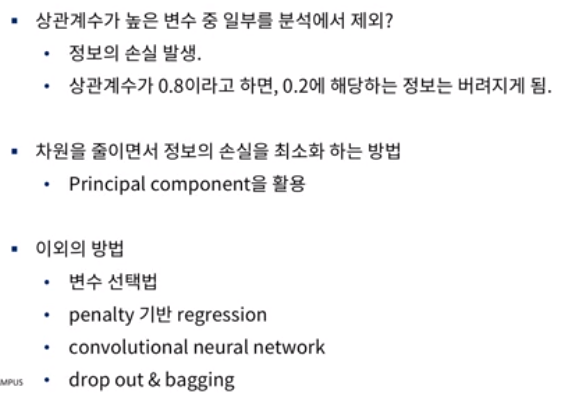
차원의 저주 : 각변수의 50%영역에 해당하는 자료를 가지고있을 때 전체자료의 얼마만큼을 확보할 수 있는가? 차원이 늘어남에 따라서 전체영역대비 설명가능한 부분이 줄어든다.

1차원에서는 50%, 2차원에서는 25%, 3차원에서는 12.5% …….

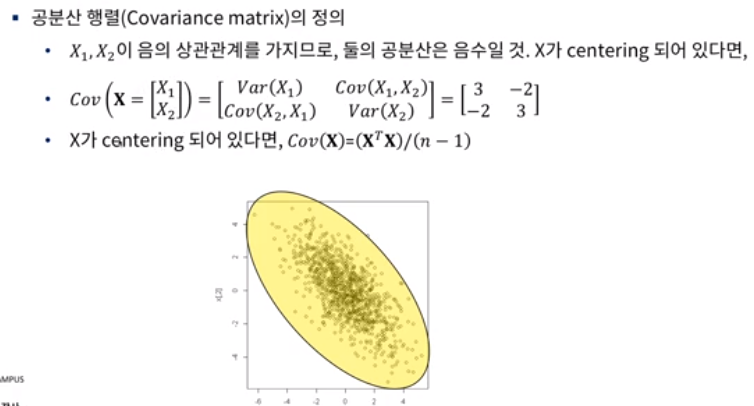
 점점 sparse해진다

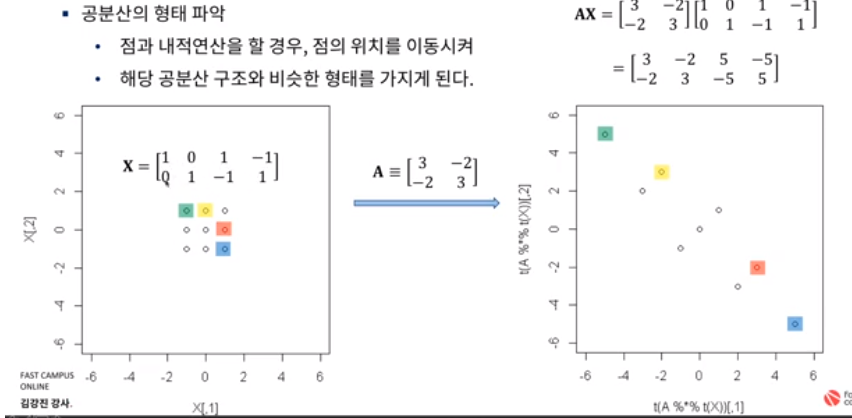
차원의 저주를 막기

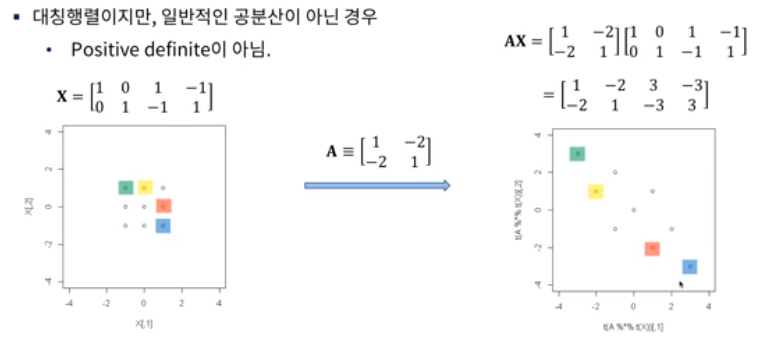
위해서는?

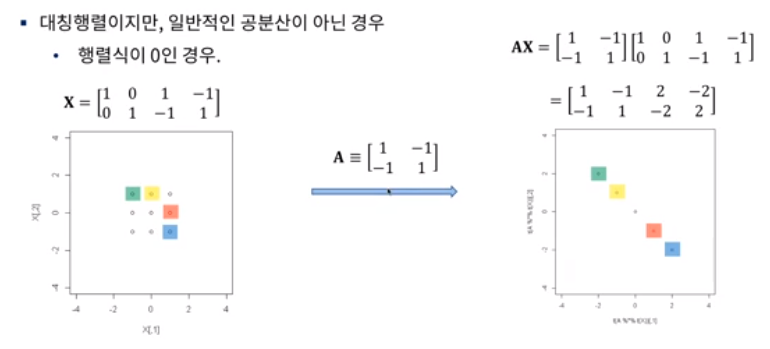


**<공분산 행렬의 이해>**



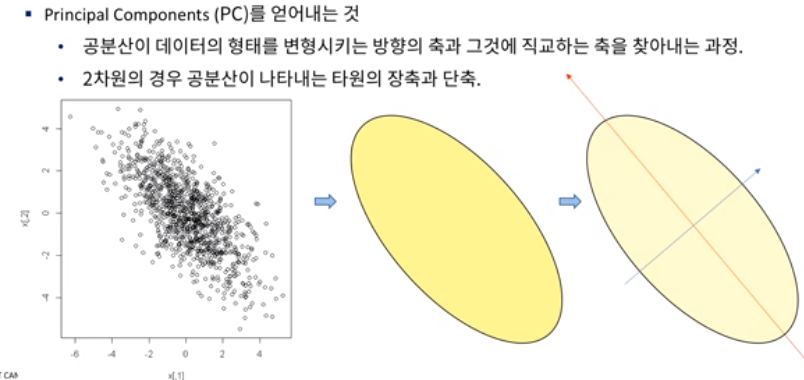


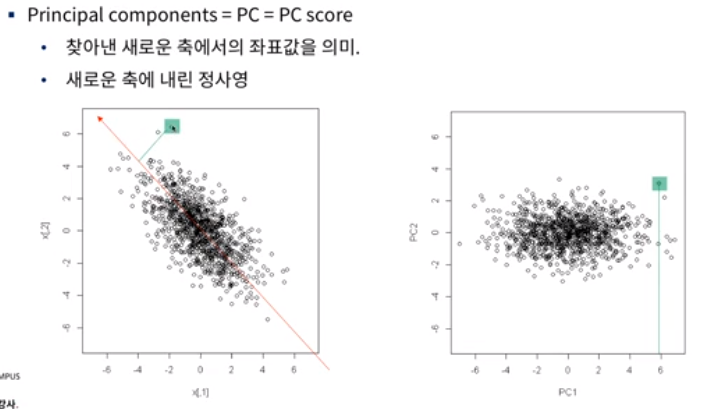




늘린형태가 아닌 대칭이거나 하나의 선으로 되는 형태 – 일반적인 공분산이 아님

**<PCA의 이해> 주성분(principal conponents)**

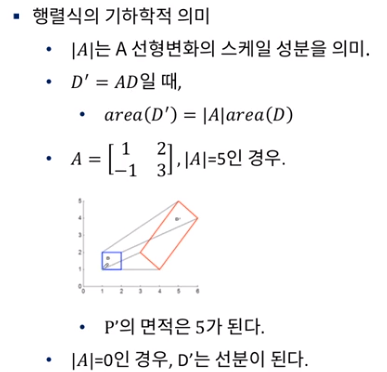




새로운 좌표값으로 표현했다 라고 생각

역행렬의 존재성을 알기 위해 determinant를 구할줄 알아야한다.

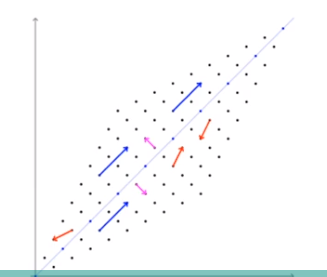
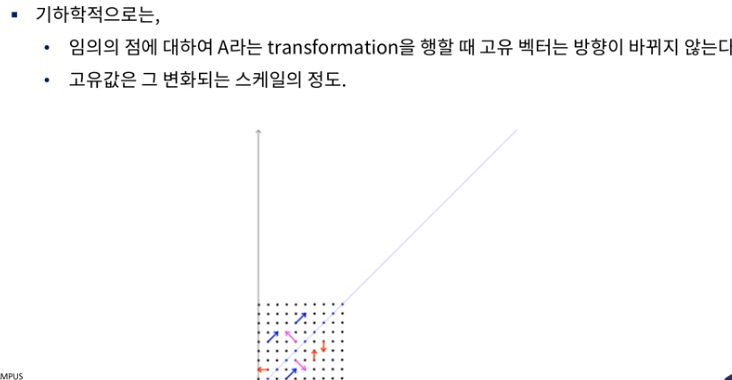
Cramer’s rule을 통해 n\*n matrix에서 determinant를 구한다.

 선형변화된 면적이 det(A)만큼 배 된다.

**<Eigen vector, eigen value>**

어떤 벡터를 선형변환 했는데 방향이 유지가 되는 벡터가 고유벡터

벡터가 얼마나 늘어났는지를 보이는게 고유값



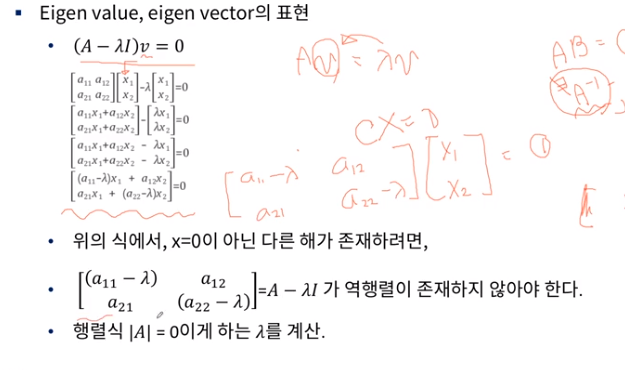
A변환 : 각 점들을 이동시키는 선형변환

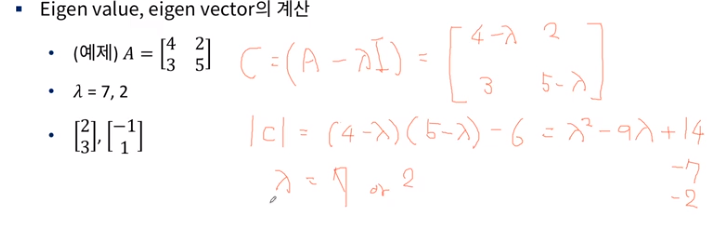
파랑과 분홍은 A라는 선형변환에도 방향이 바뀌지않는다. : 고유벡터

빨강은 방향이 바뀐다. : 고유벡터가 아니다

파랑은 A라는 선형변환에 대해 크기가 많이 변한다 : 고유값이 크다. 분홍색은 거의 변화가없다: 고유값이 작다.

고유벡터와 고유값의 계산



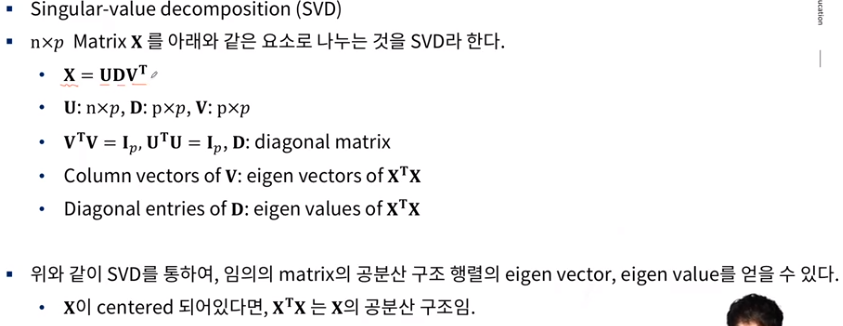


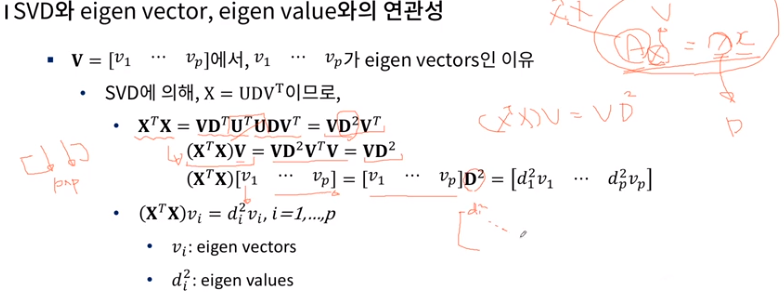
고유값을 통해 고유벡터를 구한다. (7, 2) - 고유벡터는 방향만이 의미가있기 때문에 스케일은 의미없다.

**<Singular Value Decomposition(SVD)> 어려운부분 ㅠ**

Svd를 하는이유는 계산에 있어서 훨씬 간단해진다.

U, D, V로 나눌수있다. V와 U는 orthogonal하다

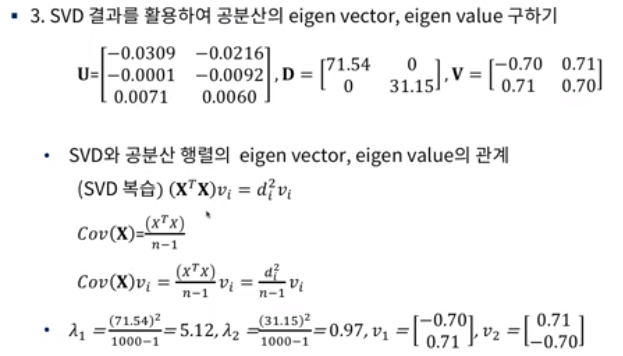


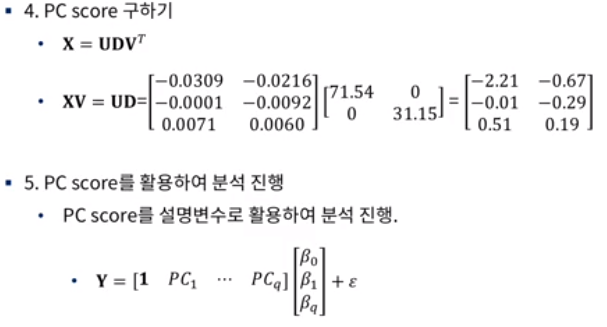


D역행렬과 D의 곱이 왜 D제곱이 되는지 알기

n개의 관측지와 p개의 변수가 있을 때 A가 공분산 구조라면 eigen vector와 eigen value는 p개가 나온다( 변수의 개수와 동일하게 나온다 ) eigen value가 클수록 거기에 대응되는 eigen vector가 중요하다 이는 자료의 분산을 굉장히 많이 설명한다. 그래서 분산을 많이 설명하는 일부만 뜯어내서 분석을 할 수 있다.

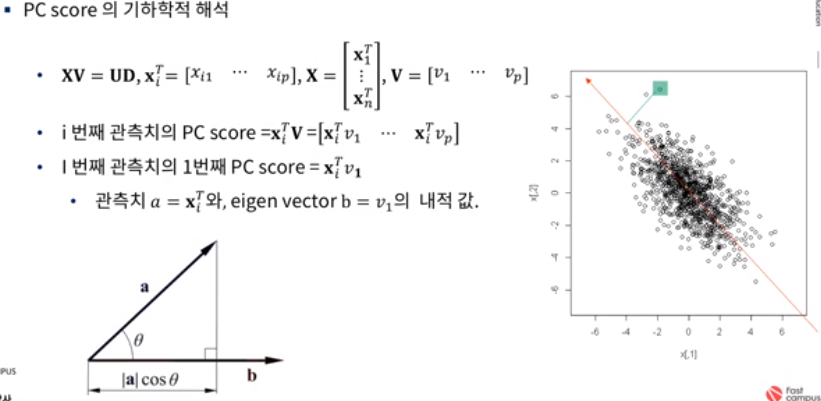
**<PCA 수행과정 >**





Eigen value가 클수록 중요한 PC가 된다. PC를 몇 개 선택해야 하는지는 실습 때 진행

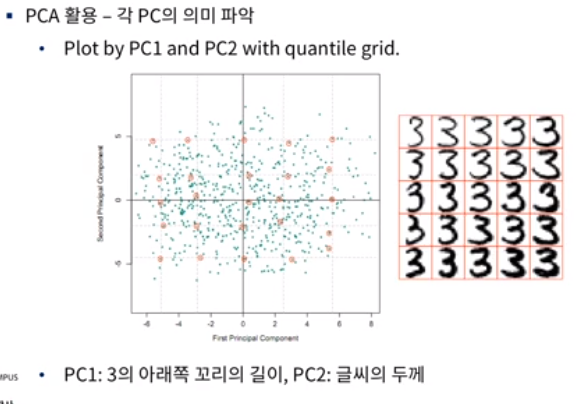
**<PCA의 심화적 이해>**



PCA의 단점은 뽑아낸 PC가 어떤 의미를 갖는지 바로 파악이 어렵다. Ex) 키와 몸무게를 뽑았을 때 의미를 알수있지만 PC1, PC2라고 하면 한번에 알기 어렵다.

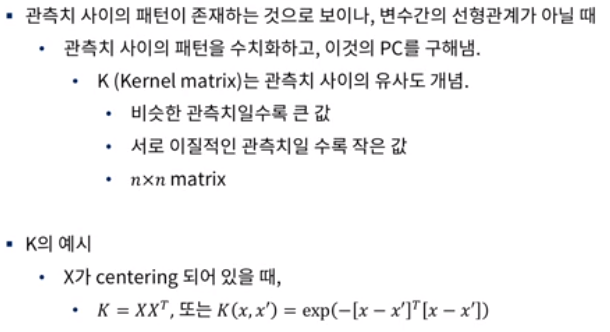
아래 예제는 PC를 시각화해서 보임으로써 PC를 바로 알아보게 구현한 모습이다.

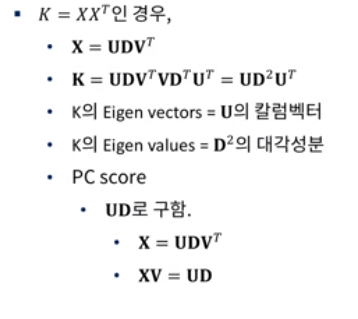
과정보다 결과해석 방법을 이해해보자

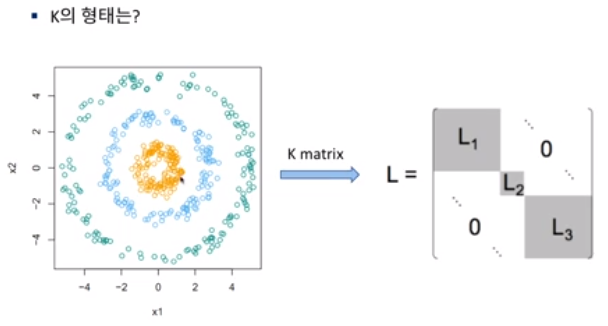


PC1이 PC2보다 분산이 크다(옆으로 넓은 분포). ( PC1이 자료에대해 큰 분산을 나타내는 변수이므로 ) -> 3인지를 알아내는 요인에 큰 영향을 끼치는게 PC1이다. 여기서는 x축이 바뀌면서 ‘3의 아래쪽 꼬리의 길이’ 가 변한다. y축이 바뀌면서 글씨의 두께가 커진다(PC2)

**<Kernel PCA> PCA가 선형관계가 아니면 잡아낼수없는 단점을 커버하기위해 나온개념**

많이 쓰이는 커널 두가지

 이전 식과 다른점은 V의 역할을 U가 하고있다. (Transpose됐기 때문에.)



노란색 부분은 가까울것이다. 유사도가 높을것이다. 유사도가 낮은 부분들은 0에 가까운 값을 띈다.

유사도가 높은 애들은 큰값이 나오고 매트릭스를 정렬했을 때 L1, L2, L3처럼 블록 형태로 모일 것이다.

* 관측치 사이의 선형적이지 않은 관계는 커널을 사용해서 (관측치사이의 고유거리를) 패턴이 있음을 나타낼수있다.