# 선형대수학팀

3팀
박이현
송지현
유종석
김민서
이주형

## 공분산행렬

### 공분산행렬

$$\Sigma_{XY} = Cov(X, Y) = E[(X-\mu)(Y-\nu)^T] = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_{nn} \end{bmatrix}$$

$$\sum X_1, X_2 = Cov(X_1, X_2) = E[(X_1-\mu_1)(X_2-\mu_2)^T] = \begin{bmatrix} Var(X_1) & Cov(X_1, X_2) \\ Cov(X_2, X_1) & Var(X_2) \end{bmatrix}$$

**정사각행렬**의 성분을 각 변수의 **분산**과 **공분산**으로 표현한 것

## 공분산의 이해

어떤 데이터들이 존재할 때 해당 데이터에 대한 분산과 공분산의 의미

데이터가 어느 방향으로, 얼만큼 많이 퍼져있는지에 대한 정보

즉, 데이터가 보유하고 있는 정보!

# 1 주성분분석(PCA)

## 주성분분석의 직관적 이해

### 주성분분석(PCA)

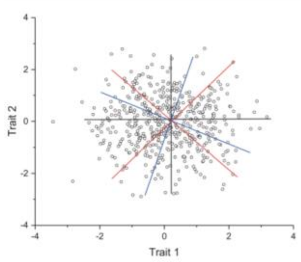
**고유값 분해**의 원리를 **공분산행렬**에 적용시킨 것

### 고유값 분해

$$A = V\Lambda V^{-1}$$

Step 1 $V$ : 축 방향 회전 → Step 2 $\Lambda$: 길이 변화 → Step 3 $V^{-1}$ : 축 방향 회전

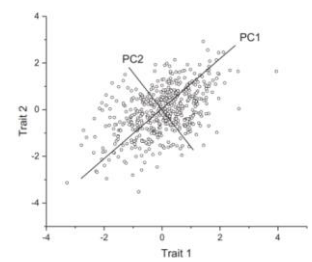
과거 데이터




선형변환

공분산행렬

현재 우리가 보고 있는 데이터




데이터가 PC1, PC2 방향으로 퍼짐

**데이터가 어느 방향으로, 얼마나 퍼져있는지 알 수 있음!**

(고유벡터) (고유값)

## 주성분분석의 장단점

**PCA의 장점**

① **고차원의 데이터를 저차원으로 축소**해 빠른 연산 가능

② 고차원 데이터에 대한 시각화 가능

**PCA의 단점**

① 저차원으로 데이터 축소하는 과정에서 **정보의 손실** 발생

② 개별 PC들이 무엇을 의미하는지에 대한 해석이 어려움

PCA를 통해서 만든 새로운 축 PC1, PC2에 대한 회귀계수 $\beta$

이것이 정확히 무엇을 의미하는지 해석하기 어려움!

# 1 주성분분석(PCA)

## Scree Plot

### Scree Plot

PCA를 통해서 차원을 축소할 때, 사용할 PC의 개수를 정하는 방법 중 **시각적으로 확인**하는 방법

**R을 통한 주성분 분석**

Prcomp로 주성분 분석 후

Summary 함수로 결과값 출력

Screeplot 함수로 plot 출력

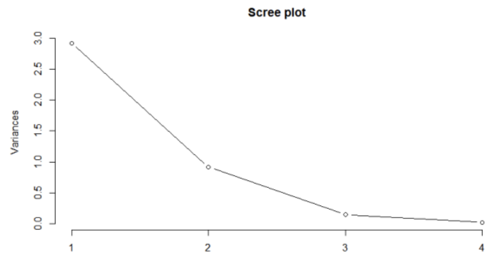
```
# 수치형 자료에만 적용 가능, center와 scale을 통해 표준화 자동 진행
PC = prcomp(iris[,-5], center = TRUE, scale = TRUE)
summary(PC)

screeplot(PC, type = "l", npcs = 4, main = "Scree plot")
```

summary(PC) 출력값

```
> summary(PC)
Importance of components:
```

| | PC1 | PC2 | PC3 | PC4 |
|---|---|---|---|---|
| Standard deviation | 1.7084 | 0.9560 | 0.38309 | 0.14393 |
| Proportion of Variance (분산값) | 0.7296 | 0.2285 | 0.03669 | 0.00518 |
| Cumulative Proportion (누적분산값) | 0.7296 | 0.9581 | 0.99482 | 1.00000 |

PC1의 분산이 0.7296, PC2가 0.2285




PC1과 PC2만으로도

**전체 데이터의 95.8% 설명 가능**

PC1, PC2 두 개로 **차원 축소**하여

PCR 진행 가능

# 1 주성분분석(PCA)

## 주성분분해의 수식적 이해

**변수추출법**

보유하고 있는 **변수들을 조합**해서

새로운 변수를 만들어내는 방법

ex) **주성분분석** (PCA), 요인분석 등

**변수선택법**

필요하지 않거나 영향력이 적은

**변수를 제외**하는 방법

ex) 전진선택법, 후진선택법 등

**기존 변수**

데이터행렬 $X$ : 관측값 $n$개, 변수 $p$개로 구성

벡터 $C$ : $p$차원의 계수벡터

$$X = \begin{bmatrix} | & | & & | \\ X_1 & X_2 & \cdots & X_p \\ | & | & & | \end{bmatrix} \qquad C = \begin{bmatrix} C_{11} & \cdots & C_{p1} \\ C_{12} & \cdots & C_{p2} \\ \vdots & \cdots & \vdots \\ C_{1p} & \cdots & C_{pp} \end{bmatrix}$$

선형결합

**새로운 변수**

$Z$

$$Z = \underline{C^T} X$$

$$Z_1 = c_{11}X_{11} + c_{12}X_{12} + \cdots + c_{1p}X_{1p}$$

선형결합을 위한 내적계산을 위해

Transpose 취함

$$Z = \begin{bmatrix} \vec{z_1} \\ \vec{z_2} \\ \cdots \\ \vec{z_p} \end{bmatrix} = \begin{bmatrix} \vec{c_1}^T X \\ \vec{c_2}^T X \\ \cdots \\ \vec{c_p}^T X \end{bmatrix} = \begin{bmatrix} \vec{c_1}^T \\ \vec{c_2}^T \\ \cdots \\ \vec{c_p}^T \end{bmatrix} X = C^T X$$

## 주성분분해의 수식적 이해

### $Z = C^T X$ 에 대한 평가

선형변환 후 데이터에 대한 정보를 얼마나 보존하고 있는가를 평가

(= 데이터의 분산)

분산이 최대가 되는 선형변환이 가장 좋은 선형변환

$(C^T X)$

$$\max_c[Var(Z)] = \max_c[Var(C^T X)] = \max_c[C^T Var(X) C] = \max_c[C^T \Sigma C]$$

데이터 행렬 $X$에 대한 분산-공분산 행렬, $\Sigma$로 표기

### $\max_c[C^T \Sigma C]$ 를 만족하려면?

① $C$가 무조건 큰 값을 가짐 → 분산이 무한히 커질 수 있으므로 제약 조건이 필요!

② $C$에 제약조건 부여 $\|C\| \rightarrow C^T C = 1$ → 일반적인 방법으로는 해결하기 어려우므로 라그랑주 승수법 사용!

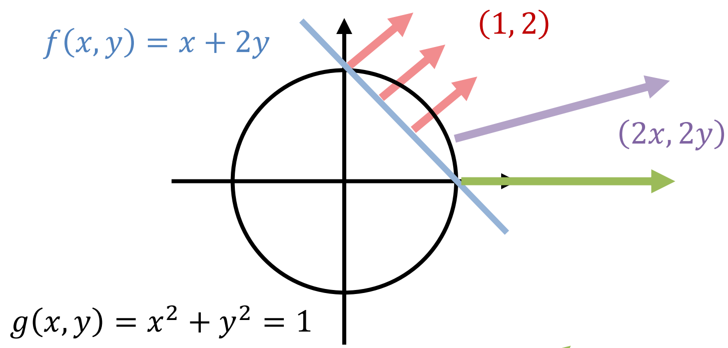
# 라그랑주 승수법

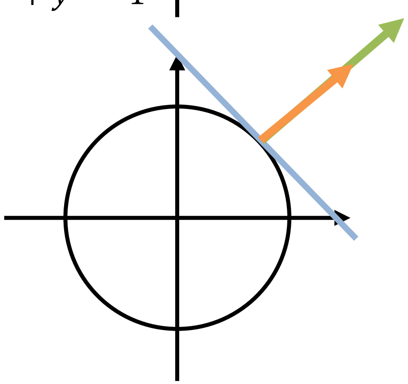
## 라그랑주 승수법의 기본가정

제약조건 $g(x,y)$ 을 만족하는 $f(x,y)$ 의 최솟값 또는 최댓값은

$f(x,y)$ 와 $g(x,y)$ 가 **접하는 지점**에 존재할 수 있음

예제) 제약조건 $g(x,y) = x^2 + y^2 = 1$을 만족하면서 $f(x,y) = x + 2y$ 직선과 접하는 최적의 점을 찾기



① 두 함수를 각각 편미분하기

편미분으로 구한 그라디언트 함수

$\nabla g(x,y) = (2x,2y)$ $\nabla f(x,y) = (1,2)$

$if\ (1,0) \rightarrow \nabla g = (2,0)$

$(0,1) \rightarrow \nabla g = (0,2)$

해당 위치에서 나가는 변화율 벡터

즉, **그라디언트**

② 두 그래프가 접하게 될 경우

$$\nabla g(x,y) = \lambda \nabla f(x,y)$$

한 함수의 그라디언트는

다른 함수 그라디언트의 상수배(λ배)가 됨

이 경우에 최적값이 될 가능성이 높음

# 1 주성분분석(PCA)

## 라그랑주 승수법과 PCA

$$f(x) \rightarrow \max_{c}[C^T \Sigma C]$$

Step 1 변화한 축 $Z$에 대해 **분산** 최대 보존

$$g(x) \rightarrow \|C\| \rightarrow C^T C = 1$$

Step 2 $C$ 에 대한 제약조건 필요

$$L = C^T \Sigma C - \lambda(C^T C - 1)$$

Step 3 라그랑주 승수법을 통한 함수 사용

$$\nabla L = \Sigma C - \lambda C = (\Sigma - \lambda)C = 0$$

Step 4 편미분을 통한 최적의 $c$값 도출

non-trivial한 $C$를 구하기 위해서는

$\det(\Sigma - \lambda I)$=0을 만족해야 함!

**목적:** 초기 변수 X들의 **선형결합**을 통해 **새로운 변수 Z**를 생성

↓

변수 Z 형성시 **분산(=데이터의 정보)을 최대로 보존**하기를 바람

↓

분산을 최대로 보존 시 **제약 조건** $C^T C = 1$ 을 걸어주어야 함

↓

조건을 만족하고 분산을 최대로 하기 위해 **라그랑주 승수법** 사용

최적의 선형결합을 만드는 $C$= **Σ의 고유벡터**

각 고유벡터들의 고유값 = **Σ의 고유값**

PC = 공분산 행렬의 **고유벡터**

**PC의 고유값** = PC가 가지고 있는 **정보의 양 (=분산의 양)**

# 편미분

### 편미분 (Partial Derivative)

**다변량 함수**에 관한 미분 방법, **특정 변수**에 대해서만 미분

미분의 대상이 되는 변수를 제외한 나머지를 모두 **상수** 취급

### 전미분 (Total Derivative)

**전체 변수**에 대한 미분, 모든 변수에 대한 **편미분 값의 합으로 표현**

### 그라디언트 (Gradient)

각 변수에 대해 모두 **편미분**을 한 후 **벡터 형태**로 표현한 값

$$\nabla f(x,y,z) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right)$$

**그라디언트 = 특정 위치에서의 변화율**

각 변수 $x, y, z$ 방향으로 얼만큼 변화하는지에 대한 변화율을 모두 나타낸 것

# 최적화

Lasso Regression

$$\min_{\beta_0,\beta}\{\sum_{i=1}^{N}(y_i-\beta_0-x_i^T\beta)^2\} \ \ subject\ to\ \sum_{j=1}^{p}|\beta_j|\le t$$

**목적함수를 최소화**하기 위한 최적의 $\beta_0, \beta$ 찾기

## 최적화 (Optimization)

목적함수를 **최대화** 혹은 **최소화**하는 파라미터 조합을 찾는 과정

ex) 경사하강법, Adam 등

## 경사하강법(Gradient Descent)

그라디언트가(Gradient) **감소**하는 방향으로 **이동(Descent)**하는 최적화 알고리즘

$$\theta = \theta - \eta \cdot \nabla \mathcal{J}(\theta)$$

$\theta$는 각각의 파라미터를 의미, 최솟값을 찾아줘야 하는 변수

$\eta$는 학습률을 의미, 한번 이동할 때의 보폭

$\mathcal{J}(\theta)$는 목적함수를 의미

## 경사하강법의 문제점

### 적절한 학습률 설정의 어려움

경사하강법을 적용할 때 **학습률**(보폭 크기, $\eta$)을 정하는데,

어떤 학습률이 가장 좋은지 알 수 없음

### 극소점에 갇힐 수 있음

맨 아래에 있는 **최소값(Global Minimum)**을 찾아가야 함

그러나 중간의 극소점(Local Minimum)을 최소값이라고 인식해서

**극소점에 갇히게 됨**

## Adam

### Adam(Adaptive Moment Estimation)

경사하강법 알고리즘 +

학습을 진행하면서 **개별 변수**마다 **다른 학습률**을 적용

**첫번째 알고리즘**

보폭 크기를 서로에게 맞춘 경우

**두번째 알고리즘**

모든 변수들은

업데이트 되는 빈도와 속도가 상이

↓

학습을 진행하면서

**개별변수**마다 **다른 학습률** 적용

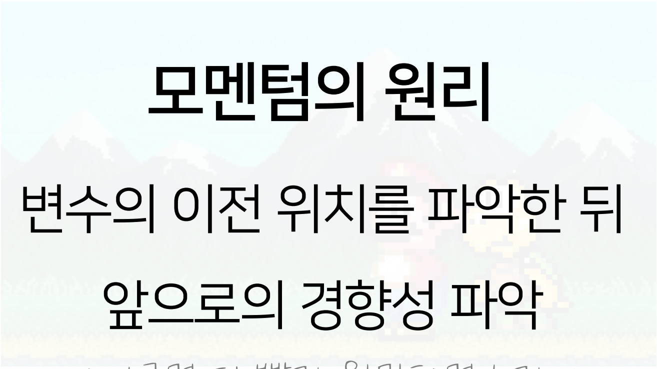
## Adam

### Adam(Adaptive Moment Estimation)

경사하강법 알고리즘 +

학습을 진행하면서 **개별 변수**마다 **다른 학습률**을 적용

첫번째 알고리즘



**모멘텀의 원리**

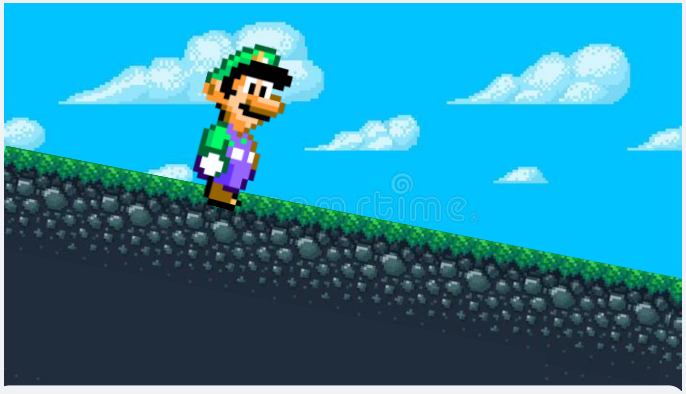
변수의 이전 위치를 파악한 뒤

앞으로의 경향성 파악

가파르면 더 빨리, 완만하면 느리게

보폭 크기를 서로에게 맞춘 경우

두번째 알고리즘



방향을 지속하는 경우

딥러닝 1주차 클린업 참고

## Adam

# Adam(Adaptive Moment Estimation)

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) {g_t}^2$$

$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \widehat{m}_t$$

### Adam의 특징

학습률을 스스로 조절하기 때문에

따로 학습률을 설정하는 번거로움을 덜 수 있음

모멘텀의 개념을 적용해 효율적으로 움직임

최근 딥러닝 분야에서 많이 활용중인 모델