선형대수학 클린업 3주차

클린업 마지막 주차인 만큼, 여러분의 니즈를 반영해서 최대한 많은 특강을 담아 두었습니다. 1~3주차 때 했던 특강들을 참고해서 주제분석 및 앞으로 남은 패키지를 잘 해결할 수 있길 바랍니다. 그동안 3주동안 정말 고생 많으셨고, 앞으로 조금만 더 같이 고생하면 끝이 보일거에요. 파이팅!!

[Week3 - Contents]

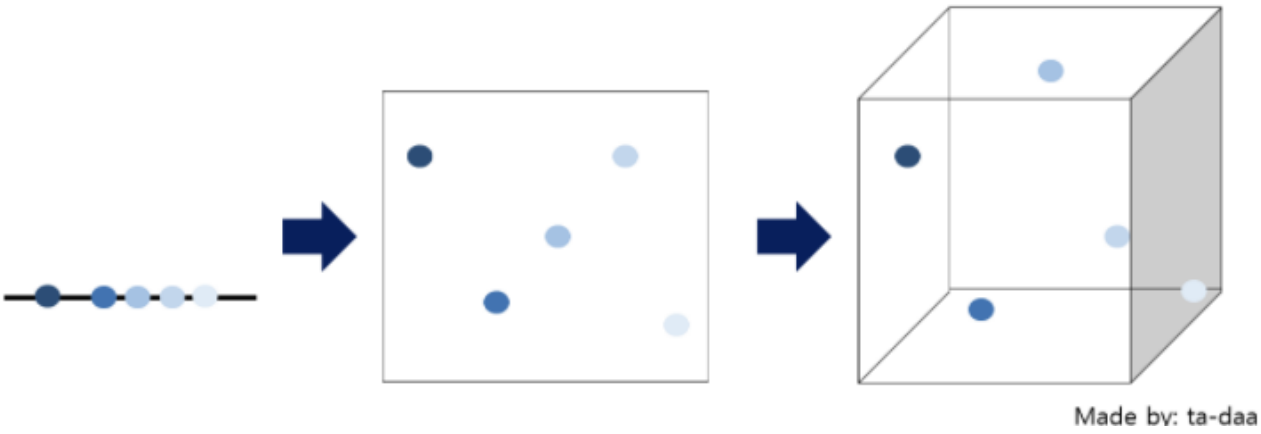
1. 주성분 분석 (Principal Component Analysis; PCA)
    1) 차원의 저주 및 차원 축소
    2) 공분산에 대한 선형대수에서의 해석(Linear Algebra in Covariance)
    3) 주성분 분석(PCA)
2. 벡터 미적분학(Vector Calculus)
    1) 편미분(Partial Differentiation)
    2) 기울기(Gradients)
    3) 벡터함수의 그라디언트(Gradients of Vector-Valued Functions)
3. 벡터 미적분학의 응용
    1) 자코비안
    2) Back Propagation
    3) 경사하강법(Gradient Descent)
4. 선형대수학팀의 3주차 특강
    1) 정규표현식 in Python
    2) PCA in Python
    3) Deep Learning in Python

1. 주성분 분석 (Principal Component Analysis; PCA)

1) 차원의 저주 및 차원 축소

■ 차원의 저주

데이터를 다룰 때 차원은 보통 feature의 개수를 가리킨다. 즉, input 데이터에서 X변수에 해당하는 것들의 개수를 의미한다. 차원이 늘어난다면 이는 곧 설명변수가 늘어나는 것을 의미하는데, 이는 데이터를 표현할 수 있는 더 잘 설명해서 분석에 도움이 되는 긍정적인 영향을 끼칠 수 있다. 하지만 역설적으로, 차원이 어느 정도 이상 증가하면 성능이 저하되는 현상이 발생하는데, 이를 **차원의 저주**라고 부른다. 차원이 너무 많아지면 연산량이 증가함에 따라 데이터의 학습 속도가 떨어지는 뿐 아니라, 데이터의 밀도가 급격하게 줄어들어 성능까지 저하되는 것이다. 자세한 내용은 데이터마이닝팀이 1주차에서 다뤘으므로 그 부분을 참고하길 바란다.



출처: [빅데이터] 차원의 저주(The curse of dimensionality) (tistory.com)

위의 그림처럼, 차원이 증가하면 밀도가 줄어들게 되어 어느 순간 이상에서 급격하게 성능저하로 이어진다.

■ 차원 축소

그렇다면, 우리는 차원의 개수를 적절히 조절하는 것이 필요하다. 물론 처음부터 우리가 스스로 변수를 수집한다면 이를 최대한 고려하면서 변수를 가져갈 수 있지만, 때때로 우리의 뜻대로 흘러가지 않는 경우도 있다. 예를 들면, 방학세미나에서 갑자기 200개의 변수가 튀어나온다던가(200개면 양반일 수도 있다), 공모전을 갔는데 뜨악 할 만큼 많은 변수가 과제로 주어진다거나 등등…. 이럴 때 우리는 차원을 축소하는 것을 고려해보게 된다. 차원 축소에는 여러가지 방법이 존재하는데, 우리는 오늘 이 중에서 **PCA(Principal Component Analysis)**라는 방법에 대해 살펴볼 예정이다.

2) 공분산에 대한 선형대수에서의 해석(Linear Algebra in Covariance)

■ 공분산(Covariance)과 분산-공분산 행렬(Variance-Covariance Matrix)

공분산은 어떤 두 확률변수의 선형관계에 대한 정보를 알려주는 척도로, 다음과 같이 정의된다.

$Cov(X, Y) = E[(X-\mu_x)(Y-\mu_y)] = E(XY) - E(X)E(Y)$

공분산은 위에서 언급했듯이 선형관계에 대한 정보를 알려주는데, 양의 선형 관계(공분산>0), 음의 선형 관계(공분산<0), 선형 관계없음(공분산 = 0)으로 나뉜다. 자세한 내용은 생략하겠다.

공분산 행렬이란, 다음과 같이 정의된다.

$X = [\, x_1\ x_2\ \cdots\ x_n\,]^T$으로 구성된 행렬이며, x_i가 각각 벡터일 때,

$$\text{공분산 행렬 } K_{XX} = \begin{bmatrix} Cov(X_1,X_1) & Cov(X_1,X_2) & \cdots & Cov(X_1,X_n) \\ Cov(X_2,X_1) & Cov(X_2,X_2) & \dots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ Cov(X_n,X_1) & Cov(X_n,X_2) & \dots & Cov(X_n,X_n) \end{bmatrix}$$

$$= \begin{bmatrix} Var(X_1) & Cov(X_1,X_2) & \cdots & Cov(X_1,X_n) \\ Cov(X_2,X_1) & Var(X_2) & \dots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ Cov(X_n,X_1) & Cov(X_n,X_2) & \dots & Var(X_n) \end{bmatrix}$$

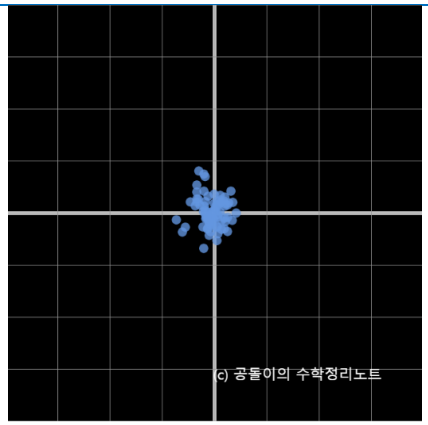
■ 분산-공분산행렬의 기하학적 의미

이제 분산-공분산행렬(이하 공분산행렬)이 기하학적으로 어떤 의미를 가졌는지 살펴보자.

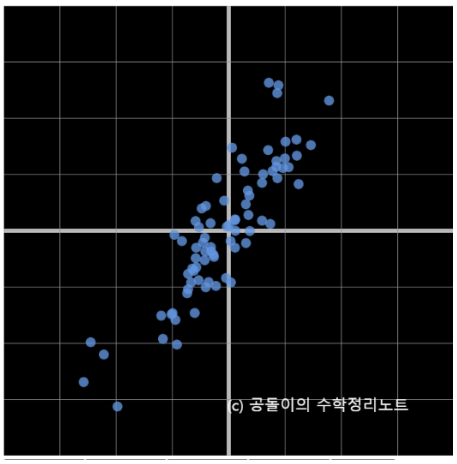
우선, 공분산이란 변수들이 어떻게 함께 움직이는지를 설명해주며, 분산은 각각의 변수의 퍼져 있는 정도를 나타내준다.

이를 바탕으로 생각해보면, 어떤 데이터에 대해 공분산 행렬을 이용해 변환하면 분산과 공분산만큼 공간이 변화하게 되고, 이는 곧 변수간에 어떤 식으로 연관되어 퍼져 있는지, 변수들이 어떻게 분포되어 있는지를 나타내게 되는 것이다.

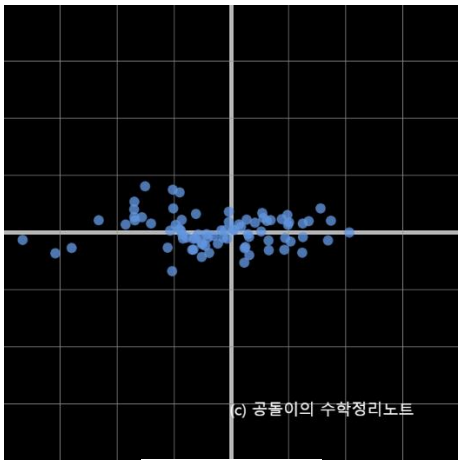
간단하게 예를 들어 알아보자



선형변환 전



$$\begin{bmatrix} 3 & 2 \\ 2 & 4 \end{bmatrix}$$



$$\begin{bmatrix} 5 & 0 \\ 0 & 1 \end{bmatrix}$$

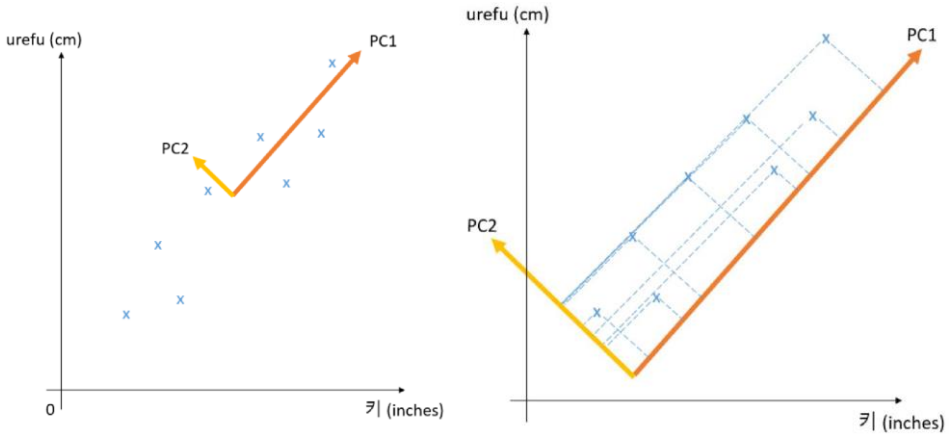
왼쪽 그래프는 x방향으로3, y방향으로는 4, x&y 방향으로는 2만큼 퍼지는 선형변환,
오른쪽 그래프는 x방향으로 5만큼만 퍼지는 선형변환의 결과이다.

3) 주성분 분석(Principal Component Analysis; PCA)

■ 주성분(Principal Component)

주성분 분석(PCA)란 변수 간의 상관관계가 존재하는 다차원의 데이터를 효율적으로 저차원의 데이터로 요약하는 방법이다. 이는 다차원의 데이터에서 그 데이터를 가장 잘 설명해주는 주성분을 찾아내 그 주성분이 이루는 공간으로 데이터를 정사영시켜서 차원을 축소하는 방법이다. 즉, PCA에서는 바로 이 주성분(PC)를 찾는 문제가 관건이 된다.

주성분 분석에서 주성분은 데이터를 각 주성분에 대해 정사영 했을 때 더 많이 퍼져있는 PC를 고르게 되는데, 즉 그 분산이 더 큰 주성분을 고르게 된다.



위 그림에서 분산이 더 큰 PC1을 PC2보다 우선적으로 고르게 된다.

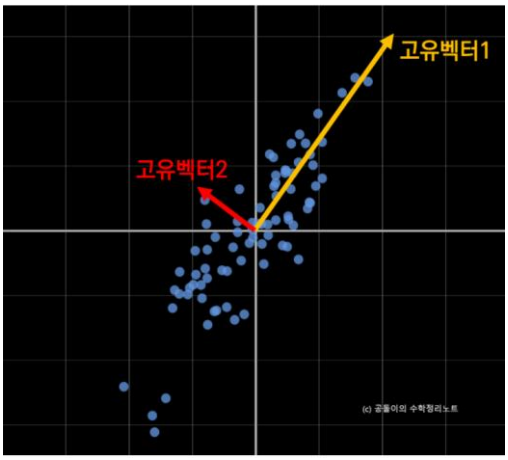
이 PC들을 구하는 방법에는 공분산행렬의 고유 값과 고유벡터가 이용되는데, 이는 고유벡터의 성질 때문이다.

지난 시간에 고유벡터는 선형변환 후에도 방향은 변하지 않고 크기만 변한다고 언급했다. 이러한 성질에 의해, 고유벡터는 선형변환의 고정된 축으로도 볼 수 있다. 또한, 여기서 고유값은 고유벡터 방향으로 변한 크기를 의미하는데, 이는 공분산행렬의 PC가 데이터에 대해 얼마나 퍼져있는지로도 해석이 가능해진다.

다음을 보자.

아래는 공분산행렬 $\begin{bmatrix} 3 & 2 \\ 2 & 4 \end{bmatrix}$의 고유값과 고유벡터이다.

```
> eigen(x)
eigen() decomposition
$values
[1] 5.561553 1.438447
```



공분산행렬의 고유값을 구한 결과는 왼쪽과 같다.
이 때, 고유벡터 1의 고유값은 5.56으로 고유벡터2(1.44)보다 많은 데이터에 대해 분포하는 것을 볼 수 있다.

즉, 공분산행렬의 고유벡터를 PC로 삼고, 이 때 고유값의 크기에 따라 그 중요성을 확인해가면서 PC들을 구할 수 있다.

■ 주성분의 개수

PCA를 통해 차원축소를 할 때에는 몇 번째 주성분까지 추출할 지 결정해야한다. 이는 PCA 결과에서 Summary를 근거로 주성분의 개수를 정하게 된다. 누적비율을 통해 몇 번째 PC까지 사용하면 데이터의 변동을 어느정도 설명할 수 있는지 확인할 수 있어서 이를 근거로 개수를 추출하는데, 일반적으로 90% 이상이 되도록 주성분으로 선택한다.

예를 통해 더 자세히 살펴보자.

```
## Importance of components:
##                           PC1     PC2    PC3     PC4     PC5      PC6
## Standard deviation     2.3935 0.44457 0.2522 0.08921 0.04529 1.49e-15
## Proportion of Variance 0.9548 0.03294 0.0106 0.00133 0.00034 0.00e+00
## Cumulative Proportion  0.9548 0.98773 0.9983 0.99966 1.00000 1.00e+00
```

위의 결과에서 PC1까지의 누적비율(Cumulative Proportion)이 0.9548, PC2까지의 누적비율이 0.98773인데, 이는 PC1이 데이터의 변동을 95.48% 만큼 설명하고 PC2까지 선택하면 데이터의 변동을 98.773%만큼 설명하는 것을 의미한다. 즉, 위의 데이터에서는 PC1만 선택해도 충분하다는 결론을 내릴 수 있다.

사실, PCA는 깊게 설명하면 한없이 깊은 이론과 증명을 맛볼 수 있을 정도로 굉장히 심오한 이론이다. 실제로, PCA를 완성할 때 처음부터 공분산행렬의 고유값과 고유벡터를 기준으로 이를 정렬하자는 발상에서 나온 것이 아니라, 먼저 Orthonormal Basis들의 집합을 정의한 후, 그 기저벡터들 중 데이터와의 Loss를 가장 적게 계산할 수 있는 기저벡터들을 우선적으로 추출하는 것에서부터 PCA의 수식이 유래됐는데, 이 수식을 정리하면 결국 공분산과 고유값과 고유벡터로 설명가능한 집합이 되는 것을 확인해 지금 이렇게 쓰이고 있다. 실제로는 정말 어마무시한 수식이 PCA를 설명하는데 쓰인다. 우리는 PCA의 원리에 대해 알고, 또 어떻게 활용하는지에 초점이 맞춰지는 게 옳은 방향이라고 생각이 되었고, 따라서 이 많은 수식을 교안에 넣고 설명하기에는 굉장히 부적절하다고 생각이 되었다.

따라서, PCA의 설명은 이정도에서 마무리하겠다.

2. 벡터 미적분학(Vector Calculus)

먼저, 엄밀하게 말해서 벡터 미적분학은 선형대수학에 포함되지 않는다. 말그대로 "미적분학"에 포함되는 요소이다. 하지만, 통계학에 관련한 책에서 선형대수학과 벡터 미적분학 관련 내용을 경계 없이 담고 있는 책들을 종종 볼 수 있다. 이는 아무래도 벡터 미적분학에서 통계학에 필요한 내용을 다루기 위해 하나의 책으로 써 내기에는 너무 양이 적고, 그렇다고 아예 다루지 않기에는 통계에서 심화된 내용을 다룰 때 필요한 내용이 포함되어있기 때문일 거라 생각이 된다. 또한, 이 역시 벡터와 행렬을 심화적으로 다루며 그 과정에서 기하학적으로 해석할 때 미적분과 선형대수의 경계가 애매한 부분도 있기 때문일 수도 있다. 우리 역시 이와 같은 이유로 벡터 미적분학을 우리 교안에도 넣기로 결정하였다.

1) 편미분(Partial Differentiation)

■ 편미분의 개념

편미분(Partial Differentiation)은 어떤 다변수함수에 대해 그 중 하나의 변수에 주목하고 나머지 변수의 값을 고정시켜 놓고 그 변수로 미분하는 것을 가리킨다.

예를 들어, x와 y로 구성된 2변수 함수 f(x,y)가 있을 때 y를 상수로 보고 이것을 x로 미분하는 일을 x로 편미분한다고 한다. 또한, 편미분 기호로는 ∂를 사용한다.

편미분은 다변수함수에서 1개의 변수의 값이 변화할 때의 변화율을 알기 위해 사용된다.

수식적으로는 다음과 같은 의미를 가지는데, 이는 하나의 변수에 주목한다는 점을 제외하고는 일반적인 미분과 정의를 같이한다.

$$\frac{\partial f}{\partial x_1} = \lim_{h\to 0}\frac{f(x_1+h, x_2, \dots, x_n) - f(x)}{h}$$

$$\frac{\partial f}{\partial x_2} = \lim_{h\to 0}\frac{f(x_1, x_2+h, \dots, x_n) - f(x)}{h}$$

$$\vdots$$

$$\frac{\partial f}{\partial x_n} = \lim_{h\to 0}\frac{f(x_1, x_2, \dots, x_n+h) - f(x)}{h}$$

■ 편미분의 계산

식이 주어진 경우 편미분을 계산하는 방법은 매우 간단하다. 미분하는 변수 외에 다른 변수는 상수 취급하고 미분을 진행하면 된다.

예를 들어, $f(x,y) = x^2 + xy$의 경우 $\frac{\partial f}{\partial x} = 2x + y$가 된다.

2) 기울기(Gradients)

■ 미분과 Chain Rule

편미분은 다변수함수에서 그 함수를 구성하는 특정 변수 하나가 변화했을 때의 변화율을 나타낸다고 위에서 언급했다.

그런데, 때때로 다변수함수 f를 구성하는 x, y, z, …가 다시 다른 변수 t에 대해 정의되어 있는 경우가 있다.
그 때, 변수 t에 대한 f의 도함수는 편미분을 사용해서 다음과 같이 정의할 수 있다.

$$\frac{df(x,y,z,\dots)}{dt} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial f}{\partial z}\frac{\partial z}{\partial t} + \cdots$$

이때, 각각의 항을 우리는 "Chain Rule"이라고 정의한다.

이 미분식은 흥미롭게도 행렬의 곱연산을 이용하여 표기할 수도 있는데, 그 내용은 다음과 같다.

함수 f가 $x_1(t),\ x_2(t)$로 구성되어 있을 때, t에 관한 f의 도함수는 다음과 같다.

$$\frac{df}{dt} = \begin{bmatrix}\frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2}\end{bmatrix}\begin{bmatrix}\frac{\partial x_1(t)}{\partial t} \\ \frac{\partial x_2(t)}{\partial t}\end{bmatrix} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial t}$$

위와 같은 경우는 f를 구성하는 $x_1,\ x_2$가 한 개의 변수 t로 이루어졌을 때의 예이다.

한 단계 더 나아가서, 다음과 같은 경우를 보자.

함수 f가 가 $x_1(s,t),\ x_2(s,t)$로 구성되어 있을 때, s, t에 대한 f의 도함수는 각각 다음과 같다.

$$\frac{\partial f}{\partial s} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial s} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial s}$$

$$\frac{\partial f}{\partial t} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial t}$$

이를 행렬로 정리하면 다음과 같이 할 수 있다.

$$\begin{bmatrix}\frac{\partial f}{\partial s} & \frac{\partial f}{\partial t}\end{bmatrix} = \begin{bmatrix}\frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2}\end{bmatrix}\begin{bmatrix}\frac{\partial x_1}{\partial s} & \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial s} & \frac{\partial x_2}{\partial t}\end{bmatrix}$$

또한, $\begin{bmatrix}\frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2}\end{bmatrix} = \frac{\partial f}{\partial \vec{x}}$, $\begin{bmatrix}\frac{\partial x_1}{\partial s} & \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial s} & \frac{\partial x_2}{\partial t}\end{bmatrix} = \frac{\partial f}{\partial(s,t)}$라고 표기하고 이를 다시 행렬로 다음과 같이 표기할 수 있다.

$$\begin{bmatrix}\frac{\partial f}{\partial s} & \frac{\partial f}{\partial t}\end{bmatrix} = \frac{\partial f}{\partial \vec{x}}\frac{\partial f}{\partial(s,t)}$$

이런 방법으로 고차원에 관한 Chain Rule도 다음과 같이 행렬로 표현할 수 있는데, 본 교안에서는 자세히 다루지 않겠다.

$$\begin{bmatrix}\frac{\partial f}{\partial s} & \frac{\partial f}{\partial t} & \cdots\end{bmatrix} = \frac{\partial f}{\partial \vec{x}}\frac{\partial f}{\partial(s,t,\cdots)}$$

■ 기울기(Gradients)

우리는 이변수함수 x,y에 대해 기울기(Gradients; 그라디언트)라는 개념에 굉장히 익숙하다. 여기서 말하는 기울기 역시 우리가 알고 있는 그 기울기와 같은 의미를 가진다. 하지만, 이변수함수의 기울기는 머리속에서 쉽게 그려지는 반면, 3차원이상의 함수에 대해서는 기울기가 어떻게 되어있는지 쉽게 감을 잡을 수 없고, 대부분의 사람은 처음 들어본 개념일 수

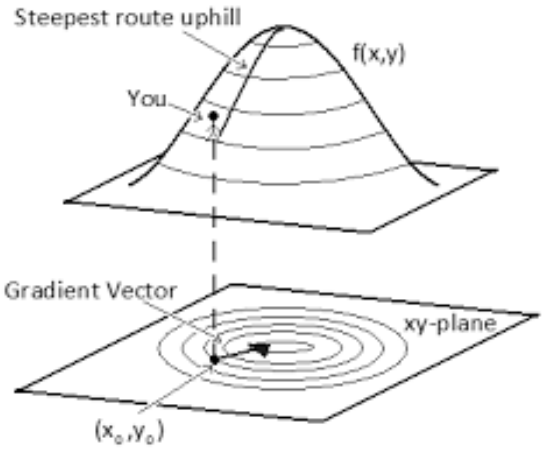
도 있다. 따라서, 이번 챕터에서는 다변수함수에서 기울기(이하 그라디언트)가 어떤 식으로 정의되는 지에 대해 살펴보겠다.

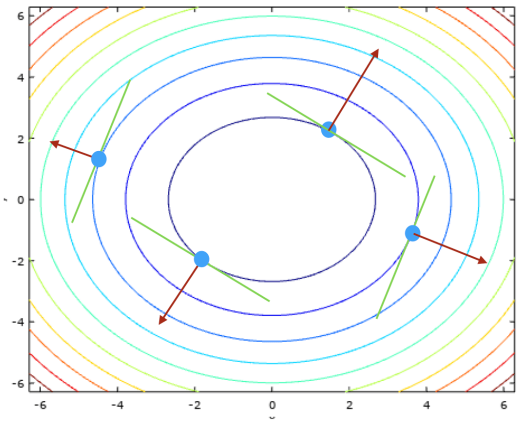
어떤 함수 f가 다중 변수 $x_1, x_2, x_3, \ldots$로 구성되어 있을 때 f의 그라디언트는 각 변수에 대하여 f를 편미분 한 것을 행(또는 열)으로 나열한 것으로 표현한다. 이 때, 그라디언트는 ▽f 혹은 grad f로 표기한다.

즉, $x_1, x_2, x_3, \ldots$로 구성된 어떤 함수 f의 그라디언트 ▽f = grad f = $\left[\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \frac{\partial f}{\partial x_3} \quad \ldots\right]$ 이다.

그라디언트에서 중요하게 생각할 점은 어떤 스칼라 함수 f에 대해서 **그라디언트 ▽f의 방향이 f가 가장 빨리 증가하는 방향을 가리킨다는 것**이다. 또한, 이는 곧 f에 대한 등고선을 그렸을 때 각 등고선에서 어떤 지점의 접선이 그 지점에서의 gradient와 반드시 수직을 이루게 됨을 나타낸다. 또한, 3차원 이상으로 확장했을 때, 어떤 점에서의 gradient는 그 점에서의 접평면과 수직을 이루게 된다.

아래의 그림을 참고하자.





보이는 것과 같이 어떤 함수 f의 등고선과 gradient는 수직, 즉 접선을 이루는 모습을 볼 수 있다.

이 성질은 매우 중요한데, 이는 이후에 우리가 다룰 Gradient Descent에 쓰이는 알고리즘의 핵심 부분이자 이유이기 때문이다. 따라서, 이를 잊지 말고 잘 기억해주기를 당부 드린다.

왜 이렇게 되는지는 다음과 같이 쉽게 증명이 가능하다.

3차원 공간을 예로 들겠다.

먼저, w = f(x,y,z) 라는 3변수 함수가 있다고 하자.

등고선이라는 것은, 어떤 함수에 대해 함수 값이 동일한 것을 모아 선으로 그려 놓은 것이다.

그러므로, 등고선을 구하기 위한 어떤 level c에 대한 교점의 집합을 f(x,y,z) = c (c는 상수)라고 표현할 수 있으며, 이를 Level Surface라고도 한다.

이제, 어떤 특정 Level Surface 값(이하 c)에 대해 함수를 살펴보겠다.

값이 c인 Level surface위의 점들을 모아 놓은 curve r(t) = <x(t), y(t), z(t)>라 하고,

g(t) = f(x(t), y(t), z(t)) = c라 하자.

즉, r은 값이 c를 가리키는 어떤 등고선 위의 점들을 나타낸 일종의 함수이다.

그렇다면, g(t) = c를 양변에 대해 미분하게 되면 Chain Rule에 의해 다음과 같이 됨을 알 수 있다.

$$\frac{dg}{dt} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt} + \frac{\partial f}{\partial z}\frac{dz}{dt} = 0$$

이를 Vector Form으로 정리하면 다음과 같다.

$$< \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} > \cdot < \frac{dx}{dt}, \frac{dy}{dt}, \frac{dz}{dt} > = 0$$

이를 표현을 바꿔보면 다음과 같이 된다.

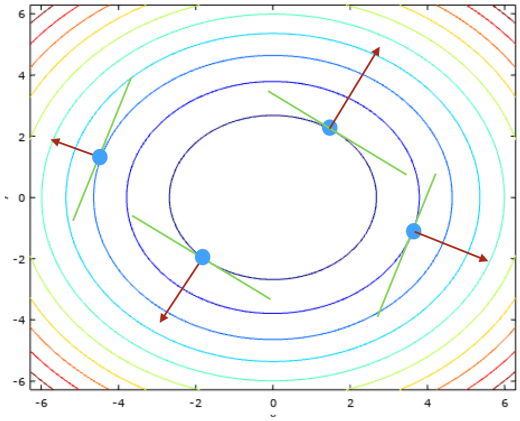
$\nabla f \cdot r'(t) = 0$

r'(t)는 c에서의 Level surface위의 curve를 뜻하는 함수 r의 접선 기울기를 말하는데,

쉽게 말하면 등고선에서의 접선의 기울기이다.

즉, 이는 곧 f의 그라디언트와 등고선에서의 접선의 기울기의 내적 값이 0이 되는 것이며, 바꿔 말하면 f의 그라디언트와 등고선의 접선이 직교 관계를 이루는 것을 말한다.

이제 그림을 다시 보자.



등고선의 접선과 f의 그라디언트가 직교를 이룬다는 말은 f의 그라디언트가 수직으로 등고선을 관통하게 된다는 것을 말하는데, 이는 쉽게 특정 점에서 가장 크게 증가하는 방향으로 그라디언트 벡터의 방향이 설정된다는 것을 알 수 있다

3) 벡터함수의 그라디언트(Gradients of Vector-Valued Functions)

■ 벡터함수(Vector-Valued Functions)

지금까지는 벡터함수에 관해서 다루기보다는 일반적인 함수(f: $\mathbb{R}^n \to \mathbb{R}$)에서 편미분 등의 개념을 설명했는데, 이제 본격적으로 벡터함수에 관해 다뤄보겠다.

우선, 벡터함수의 개념을 먼저 정립하자.

벡터함수는 다음과 같이 정의할 수 있다.

f: $\mathbb{R}^n \to \mathbb{R}^m$과 벡터 **x** = $[x_1, x_2, \cdots, x_n]^T$ 로 주어졌다고 할 때, **f(x)** = $\begin{bmatrix} f_1(x) \\ \vdots \\ f_m(x) \end{bmatrix} \in \mathbb{R}^m$으로 정의한다.

이 때, **f**는 함수 $f_1, f_2, \cdots, f_n$ 들의 벡터로 $\begin{bmatrix} f_1 \\ \vdots \\ f_m \end{bmatrix}$로 표현되며 각 f_i는 f_i: $\mathbb{R}^n \to \mathbb{R}$로 일반적인 실수 함수이다.

■ 벡터함수의 그라디언트(Gradients of Vector-Valued Functions)

벡터함수의 기본적인 개념에 대해 설명했으니 본격적으로 벡터함수의 그라디언트에 대해서 설명하겠다.

우선, 실수함수 $f_1, f_2, \cdots$에 대해 **f** = $\begin{bmatrix} f_1 \\ \vdots \\ f_m \end{bmatrix}$, 벡터 **x** = $\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$로 주어졌다고 하자.

이 때, 벡터함수 **f**를 특정 x_i에 대해 편미분하면 다음과 같이 구해진다.

$$\frac{\partial f}{\partial x_i} = \begin{bmatrix} \frac{\partial f_1}{\partial x_i} \\ \vdots \\ \frac{\partial f_m}{\partial x_i} \end{bmatrix}$$

또한, 벡터함수 **f**를 벡터 **x**에 대해 편미분하면 다음과 같이 구할 수 있다.

$$\frac{\partial f(x)}{\partial x} = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} & \cdots & \frac{\partial f(x)}{\partial x_n} \end{bmatrix}$$

그리고 위 식은 최종적으로 다음과 같이 전개된다.

$$\frac{\partial f(x)}{\partial x} = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} & \cdots & \frac{\partial f(x)}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1(x)}{\partial x_1} & \cdots & \frac{\partial f_1(x)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m(x)}{\partial x_1} & \cdots & \frac{\partial f_m(x)}{\partial x_n} \end{bmatrix}$$

바로 위의 식을 우리는 벡터함수 **f**의 그라디언트라고 정의하며, $\nabla_x \mathbf{f}$ 로 표기한다.

3. 벡터 미적분학의 응용

1) 자코비안(Jacobian)

자코비안은 행렬을 다루면서 정말 많이 만나게 되는 개념이다. 하지만, 통계학을 공부하면서 자코비안이라는 개념을 마주쳤을 때, 이 행렬에 대해서 깊게 배우지 않거나, 혹은 깊게 생각하지 않고 그냥 넘어가는 경우가 다반사였던 것 같다(필자의 수리통계학 입문 강의를 들었을 때의 경험으로는..?). 그래서 이번에는 자코비안이 어떤 행렬을 가리키고, 어떤 역할을 하는지 좀 더 자세히 들여다보고자 한다.

우리는 바로 위의 행렬을 벡터함수 **f**의 그라디언트라고 했었는데, 이를 **Jacobian**이라고도 하며, 이는 J로 표기한다.

즉, $J = \nabla \mathbf{x}\ \mathbf{f} = \frac{\partial f(x)}{\partial x} = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} & \cdots & \frac{\partial f(x)}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1(x)}{\partial x_1} & \cdots & \frac{\partial f_1(x)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m(x)}{\partial x_1} & \cdots & \frac{\partial f_m(x)}{\partial x_n} \end{bmatrix}$ 이다.

이 Jacobian은 기하학적으로 축 변환(coordinate transformation)의 역할을 하는데,

먼저 이게 어떤 식으로 이런 역할을 하는 지 예를 통해 살펴본 후, 실제로 어떻게 사용되는 지에 관해서도 살펴보겠다.

1) 어떤 식으로 축 변환을 수행하는지

$x = (x_1, x_2)$, $y = (y_1, y_2)$이며

$y_1 = -2x_1 + x_2$

$y_2 = x_1 + x_2$의 관계를 가진다고 하자.

이 때, x에서 y로의 축변환은 $J = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} -2 & 1 \\ 1 & 1 \end{bmatrix}$을 통해 할 수 있다.

즉, 임의의 $x = (x_1, x_2)$에 대해 위의 자코비안 행렬 J를 곱하게 되면 그에 대응되는 y벡터로 변하게 되며,

이는 곧 축변환을 의미한다.

예를 들어, $x = (1,1)$ 은 $\begin{bmatrix} -2 & 1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$ 를 만족해 $y = (-1, 2)$로 변환되는 것이다.

2) 수리통계학에서 이를 어떻게 활용하는 지

수리통계학에서도 축변환의 개념 차원에서 자코비안 행렬을 다루게 된다.

더 정확히는, 자코비안 행렬의 행렬식이 사용되는데, 이는 행렬식이 그 행렬의 행벡터들이 이루는 부분의 넓이나 부피 등 크기를 나타내기 때문이다. 즉, 좌표축을 변환시켰을 때, 크기의 변화를 반영한거라고 생각하면 된다.

예를 들어보자.

어떤 독립변수 X_1, X_2에 대해 $Y_1 = X_1, Y_2 = X_1+X_2$라고 새로 정의했다고 하자.

즉, $X_1 = Y_1, X_2 = -Y_1 + Y_2$가 된다.

그렇다면, $J = \begin{bmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_1}{\partial y_2} \\ \frac{\partial x_2}{\partial y_1} & \frac{\partial x_2}{\partial y_2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$이 되어, $|J| = \begin{vmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_1}{\partial y_2} \\ \frac{\partial x_2}{\partial y_1} & \frac{\partial x_2}{\partial y_2} \end{vmatrix} = \begin{vmatrix} 1 & 0 \\ -1 & 1 \end{vmatrix} = 1$이 되어

pdf $f_{Y_1Y_2}(y_1, y_2) = f_{X_1X_2}(y_1, -y_1+y_2)\,|J| = = f_{X_1X_2}(y_1, -y_1+y_2)$ 이 된다.

여기서, 왜 자코비안이 $\begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} \end{bmatrix}$가 아닌 $\begin{bmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_1}{\partial y_2} \\ \frac{\partial x_2}{\partial y_1} & \frac{\partial x_2}{\partial y_2} \end{bmatrix}$로 정의되는 지 궁금할 수 있다.

이는 간단히 말해서 적분식을 변환할 때 좌표축을 변환하는 것은 맞지만, 역변환이 일어나는 것이기 때문인데,

잘 보면 pdf가 $f_{X_1X_2}(x_1, x_2)\,|J|$가 아닌 $f_{X_1X_2}(y_1, -y_1+y_2)\,|J|$ 를 사용한 것을 볼 수 있다. 이는 f내에서 역변환이 일어나 $f_{X_1X_2}(u(y_1,y_2),\ v(y_1,y_2))$ 을 사용하는 모양을 띤다. 쉽게 생각해서, 실제로는 y_1, y_2로 이루어져 있는 함수를 X_1, X_2로 구

성된 f라는 함수안에 다시 집어넣은 모양이므로 축변환을 반영하기 위해 $\begin{bmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_1}{\partial y_2} \\ \frac{\partial x_2}{\partial y_1} & \frac{\partial x_2}{\partial y_2} \end{bmatrix}$를 곱해줘야 하는 것이다.

이를 공식 차원에서 정리해서 2차원의 경우

$\iint f(x,y)dxdy = \iint f\left(g(u,v), h(u,v)\right)|J(u,v)|dudv$로 나타남을 모두 알고 있을 것이다.

물론, 이 Jacobian을 가지고 더 고차원의 축변환도 연산이 가능하다.

하지만, 여기서는 생략하겠다.

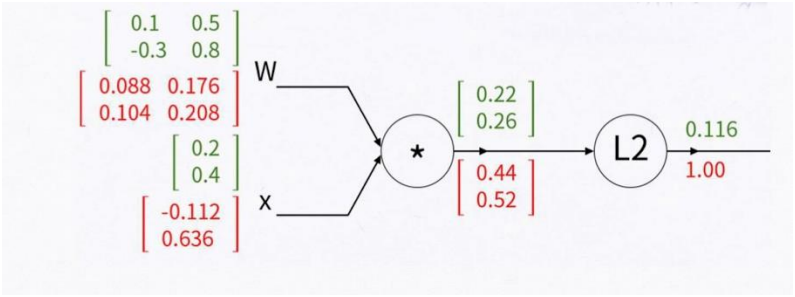
2) Back Propagation

이제 딥러닝에서 벡터 미적분학이 어떻게 적용되는지 살펴보겠다. 사실, "어떻게 적용되는지"라는 말이 굉장히 어색할 수 있는데, 그 이유는 딥러닝 이론이 사실상 선형대수학의 연속이기 때문이다. 단순히 행렬간의 계산을 넘어서 벡터에서의 미분 및 적분이 딥러닝 이론의 초석을 이루게 된다.

선형대수학 스터디에서 딥러닝에 관해 굉장히 깊게 접근하는 것은 부적절하기 때문에 본 교안에서는 간단하게 입력값이 벡터일 때의 BackPropagation에서 벡터 미적분학이 쓰이는 예시만 들고 마치겠다.

BackPropagation이란, 딥러닝에서 Gradient를 구할 때 사용하는 개념인데, 일반적으로 복잡한 비선형 함수의 Gradient는 한 번에 계산하는 것이 매우 힘드므로 이를 하나씩 연쇄적으로 구하면 더 간단해지므로 식을 뒤에서부터 한 단계씩 Gradient를 곱하는 과정을 의미한다. 선형대수학 강의인만큼 자세한 개념은 생략하며, 예를 통해 이를 수학적으로 어떻게 계산하고, 이 과정에서 벡터 미적분학이 어떻게 쓰이는지 알아보겠다.

다음 그림을 보자.



이 그림에서 *은 행렬의 곱연산을 뜻하고, L2는 input 값을 제곱하는 함수를 뜻한다.

즉, 최종 결과식은 다음과 같이 정해진다.

$f(x,W) = ||W\cdot x||^2 = \sum_{i=1}^{n}(W\cdot x)_i^2$ where L2 is defined as f

이 때, 편의상 *을 이용해 연산하는 중간 결과물을 q라고 놓겠다. 즉, q = W·x이다.

BackPropagation과정에서 우리는 $\nabla_W f$, $\nabla_x f$, $\nabla_q f$을 구하게 된다.

(i) $\nabla_q f$ 구하기

$f(q) = ||q||^2 = q_1^2+q_2^2+ \cdots + q_n^2$이므로 $\frac{\partial f}{\partial q_i} = 2q_i$이다.

즉, 모든 i에 대해서 위의 등식이 성립하므로 $\nabla_q f = 2q$라고 쓸 수 있다.

따라서, q가 $\begin{bmatrix}0.22\\0.26\end{bmatrix}$일 때, $\nabla_q f = \begin{bmatrix}0.44\\0.52\end{bmatrix}$가 된다.

(ii) $\nabla_W f$ 구하기

$\frac{\partial f}{\partial W_{ij}} = \sum_k \frac{\partial f}{\partial q_k}\cdot\frac{\partial q_k}{\partial W_{ij}}$인데, 이 때 q = W·x이므로 $\frac{\partial q_k}{\partial W_{ij}} = 1_{k=i}x_j$이다.

($\because$ q = W·x = $\begin{bmatrix}W_{1,1}x_1+\cdots+W_{1,n}x_n\\ \vdots \\ W_{n,1}x_1+\cdots+W_{n,n}x_n\end{bmatrix}$)

따라서, $\frac{\partial f}{\partial W_{ij}} = \sum_k \frac{\partial f}{\partial q_k}\cdot\frac{\partial q_k}{\partial W_{ij}} = \sum_k 2q_k\cdot(1_{k=i}x_j) = 2q_ix_j$이므로 $\nabla_W f = 2q\cdot x^T$이다.

즉, q와 x가 각각 $\begin{bmatrix}0.22\\0.26\end{bmatrix}$, $\begin{bmatrix}0.2\\0.4\end{bmatrix}$로 주어졌을 때 $\nabla_W f = \begin{bmatrix}0.44\\0.52\end{bmatrix}\begin{bmatrix}0.2 & 0.4\end{bmatrix} = \begin{bmatrix}0.088 & 0.176\\0.104 & 0.208\end{bmatrix}$가 된다.

(iii) $\nabla_x f$ 구하기

$\frac{\partial f}{\partial x_i} = \sum_k \frac{\partial f}{\partial q_k}\cdot\frac{\partial q_k}{\partial x_i}$인데, 이 때 q = W·x이므로 $\frac{\partial q_k}{\partial x_i} = W_{k,i}$이다.

따라서, $\frac{\partial f}{\partial x_i} = \sum_k \frac{\partial f}{\partial q_k}\cdot\frac{\partial q_k}{\partial x_i} = \sum_k 2q_k\cdot W_{k,i}$이므로 $\nabla_x f = 2W^T\cdot q$이다.

즉, q와 W가 각각 $\begin{bmatrix}0.22\\0.26\end{bmatrix}$, $\begin{bmatrix}0.1 & 0.5\\-0.3 & 0.8\end{bmatrix}$일 때, $\nabla_x f = \begin{bmatrix}0.2 & -0.6\\1.0 & 0.8\end{bmatrix}\begin{bmatrix}0.22\\0.26\end{bmatrix} = \begin{bmatrix}-0.112\\0.636\end{bmatrix}$ 이 된다.

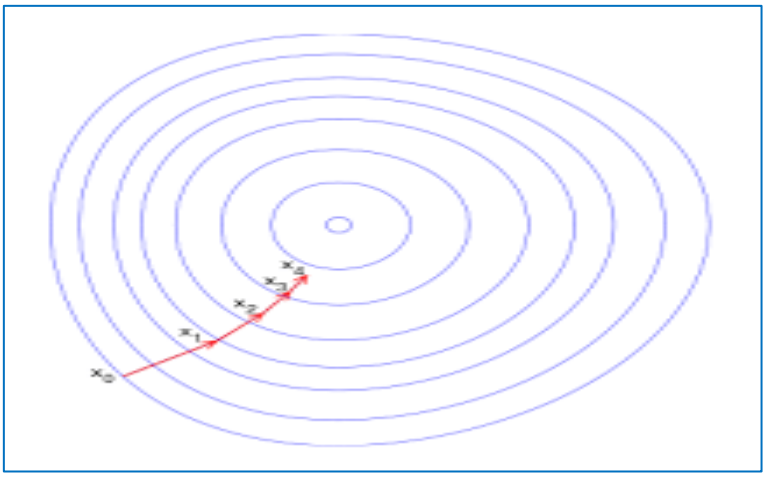
위는 딥러닝에서 벡터 미적분이 쓰이는 하나의 예일뿐이고, 딥러닝 이론을 접하다보면 이보다 훨씬 많은 부분에서 벡터 및 행렬의 미적분이 기초가 되는 것을 볼 수 있다.

3) Gradient Descent

■ Gradient Descent

Gradient Descent란 그라디언트를 이용해 최소값을 찾을 때의 방법을 일컫는다. 대략적인 알고리즘은 이렇다. 그라디언트가 가리키는 방향은 등고선에서의 접선과 수직 방향을 가리킨다고 했는데, 이는 위에서 봤듯이 등고선에서 가장 빠르게 증가 감소하는 방향을 가리킨다는 것을 알 수 있었다. 즉, 등고선에서 값이 변할 때마다 그 값에서 그라디언트가 가리키는 방향으로 움직이면 최대값이 되고, 이 과정을 **Gradient Ascent**라고 한다. 하지만, 우리가 통계에서 이 방법을 사용하는 것은 주로 Loss값을 최소화할 때 이용하므로 각 등고선 값에서 그라디언트에 (-1)을 곱해서 정확히 반대 방향으로 바꿔서 알고리즘을 운영하는 방법을 이용하는데, 바로 이 방법을 **Gradient Descent**라고 한다.

이를 그림으로 나타내면 다음과 같다.



보이는 것처럼 일정 간격마다 그라디언트 벡터에 반대방향으로 계속해서 나아가다 보면 Maximum에 도달하게 될

것이라는 발상에서 출발한 알고리즘이 Gradient Descent이다.