딥러닝을 위한 수학 I

김영록

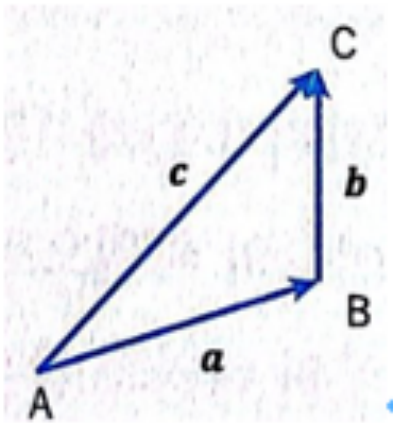
한국외대
교육대학원

2024.3.28

1. 3.2 덧셈, 뺄셈, 스칼라배
   - 3.2.1 벡터의 덧셈
   - 3.2.2 벡터의 뺄셈
   - 3.2.3 벡터의 스칼라배
2. 3.3 길이와 거리
   - 3.3.1 벡터의 길이
   - 3.3.2 $\sum$ 기호의 의미
   - 3.3.3 벡터 간의 거리
3. 3.4 삼각함수
   - 3.4.1 삼각비
   - 3.4.2 삼각함수
   - 3.4.3 삼각함수의 그래프
   - 3.4.4 직각삼각형의 변을 삼각함수로 표현하기
4. 3.5 내적
   - 3.5.1 절댓값과 내적의 정의
   - 3.5.2 벡터 성분과 내적의 공식
5. 3.6 코사인 유사도
   - 3.6.1 코사인 유사도
6. 3.7 행렬과 행렬 연산
   - 3.7.1 1 출력 노드의 내적 표현
   - 3.7.2 3 출력 노드의 행렬곱 표현
7. 모두의 딥러닝/머신러닝: 실습하기

벡터의 연산에는 벡터 간의 덧셈과 뺄셈, 그리고 스칼라배(scalar multiple)가 있습니다. 이러한 벡터 연산이 2차원에서의 '크기와 방향을 가진 양'으로는 어떻게 표현될 수 있는지 먼저 살펴보고, 이어서 성분 표기 방법으로는 어떤 모양이 되는지 살펴 보겠습니다.

- 우선 벡터의 덧셈을 알아봅시다. 벡터를 더할 때는 앞서 설명한 것처럼 '시작점과 끝점으로 표현'한다는 개념을 떠올리면 이해하기 쉽습니다. 그림 3-5와 같은 세 개의 벡터로 예를 들어 보겠습니다.



$$\mathbf{a} = \overrightarrow{AB},\ \mathbf{b} = \overrightarrow{BC},\ \mathbf{c} = \overrightarrow{AC}$$

- 벡터의 덧셈 $\mathbf{a}+\mathbf{b}$가 의미하는 것은 첫 번째 벡터의 끝점 B가 또 다른 벡터의 시작점이라고 할 때 전체적으로 어디에서 시작해서 어디로 끝나는지를 나타냅니다.
- 즉 'A에서 출발해서 B를 경유한 후, 마지막에 C에 도착한다면 전체적으로는 A에서 출발해서 C로 바로가는 것과 같다'라는 의미입니다.
- 이를 수식으로 표현하면 다음과 같고, 이 식이 바로 벡터의 덧셈을 나타냅니다

$$\mathbf{a}+\mathbf{b}=\mathbf{c} \quad \text{또는} \quad \overrightarrow{AB}+\overrightarrow{BC}=\overrightarrow{AC}$$

- 다음으로 벡터의 덧셈을 성분 표시 방법으로 표현해 봅시다.

$$\mathbf{a} = (a_1, a_2),\ \mathbf{b} = (b_1, b_2)$$

- 위와 같이 $\mathbf{a}, \mathbf{b}$가 있을 때 이 둘을 합한 $\mathbf{c}$는 다음과 같다는 것을 직관적으로 알 수 있습니다.

$$\mathbf{c} = \mathbf{a} + \mathbf{b} = (a_1 + b_1, a_2 + b_2)$$

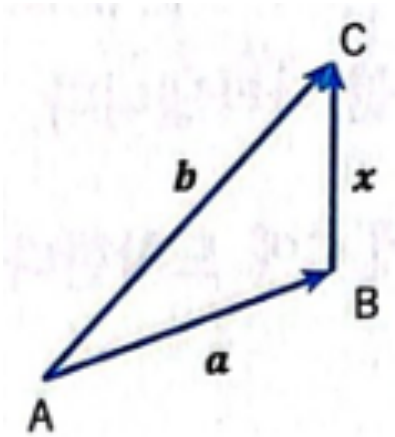
- 즉, 성분 표시 방법에서는 벡터의 덧셈이 곧 성분 간의 덧셈이 됩니다.
- 이러한 방법은 3차원에서는 물론 n차원까지도 확장할 수 있습니다. 벡터의 덧셈을 n차원의 성분 표시로 표현해서 일반화해 봅시다.

$$\mathbf{a} = (a_1, a_2, \cdots, a_n),\ \mathbf{b} = (b_1, b_2, \cdots, b_n)$$

- 위와 같이 $\mathbf{a}, \mathbf{b}$가 있을 때 이 둘을 합한 $\mathbf{c}$는 다음과 같이 표현할 수 있습니다.

$$\mathbf{c} = \mathbf{a} + \mathbf{b} = (a_1 + b_1, a_2 + b_2, \cdots, a_n + b_n)$$

- 이번에는 두 개의 벡터 $\mathbf{a}$와 $\mathbf{b}$가 있을 때 둘 간의 차이인 $\mathbf{b}-\mathbf{a}$가 어떤 모습일지 살펴 봅시다. 우선 그림 3-6과 같이 두 벡터 $\mathbf{a}$와 $\mathbf{b}$의 시작점을 똑같이 맞춥시다.



- $\overrightarrow{BC}=\mathbf{x}$라고 할 때 벡터의 덧셈으로 다음과 같은 식을 쓸 수 있습니다.

$$\mathbf{a}+\mathbf{x}=\mathbf{b}$$

- 이를 $\mathbf{x}$를 기준으로 풀어 쓰면 다음과 같이 벡터의 뺄셈으로 표현할 수 있습니다.

$$\mathbf{x}=\mathbf{b}-\mathbf{a}$$

- 이번에는 벡터의 뺄셈을 벡터의 성분 표시로 표현해 봅시다.

$$\mathbf{a} = (a_1, a_2),\ \mathbf{b} = (b_1, b_2)$$

- 위와 같이 $\mathbf{a}, \mathbf{b}$가 있을 때 이 둘 간의 뺄셈 $\mathbf{x}$는 다음과 같습니다.

$$\mathbf{x} = \mathbf{b} - \mathbf{a} = (b_1 - a_1, b_2 - a_2)$$

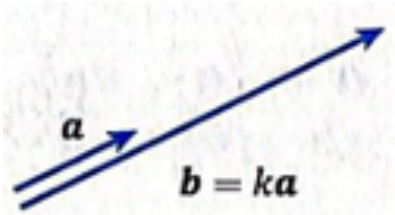
- 즉, 벡터의 덧셈과 마찬가지로 성분 표시 방법에서는 벡터의 뺄셈이 곧 성분 간의 뺄셈이 됩니다.
- 이러한 방법은 3차원에서는 물론 n차원까지도 확장할 수 있습니다. 벡터의 뺄셈을 n차원의 성분 표시로 표현해서 일반화해 봅시다.

$$\mathbf{a} = (a_1, a_2, \cdots, a_n),\ \mathbf{b} = (b_1, b_2, \cdots, b_n)$$

- $\mathbf{a}, \mathbf{b}$가 있을 때 이 둘 간의 뺄셈 $\mathbf{x}$는 다음과 같이 표현할 수 있습니다.

$$\mathbf{x} = \mathbf{b} - \mathbf{a} = (b_1 - a_1, b_2 - a_2, \cdots, b_n - a_n)$$

- 벡터의 덧셈과 뺄셈에 이어 이번에는 곱셈에 해당하는 스칼라배에 대해 알아봅시다. 사실 벡터의 곱셈에는 '내적'이라고 하는 또 다른 형태의 곱셈이 있는데 여기서 설명하기에는 다소 복잡할 수 있으니 뒤에 나올 3.5절에서 자세히 설명하겠습니다. 대신 여기서 설명하는 '스칼라배'는 '내적'에 비하면 쉽게 이해할수 있는 곱셈 개념입니다.



- 그림 3-7을 살펴봅시다. $\mathbf{a}$라는 벡터가 있을 때 그 벡터와 방향이 같고 길이가 k배인 벡터 $\mathbf{b}$를 벡터의 스칼라배라 하고 다음과 같이 표현할 수 있습니다.

$$\mathbf{b} = k\mathbf{a}$$

- 다음은 벡터의 성분 표시로 표현해 봅시다.

$$\mathbf{a} = (a_1, a_2)$$

- 위와 같은 $\mathbf{a}$가 있을 때 벡터의 스칼라배 $\mathbf{b}$는 다음과 같습니다.

$$\mathbf{b} = k\mathbf{a} = (ka_1, ka_2)$$

- 이러한 방법은 3차원에서는 물론 n차원까지도 확장할 수 있습니다. 벡터의 스칼라배를 n차원의 성분 표시로 표현해서 일반화해 봅시다.

$$\mathbf{a} = (a_1, a_2, \cdots, a_n)$$

- 위와 같은 $\mathbf{a}$가 있을 때 벡터의 스칼라배 $\mathbf{b}$는 다음과 같습니다.

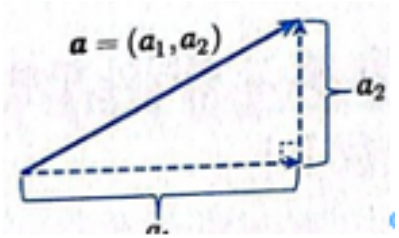
$$\mathbf{b} = k\mathbf{a} = (ka_1, ka_2, \cdots, ka_n)$$

백터를 다룰 때 중요한 양으로 '길이(절댓값)'가 있습니다. '길이'의 개념을 응용하면 두 벡터 사이의 거리를 정의할 수 있습니다. 이번 절에서는 n차원 벡터까지 고려해서 벡터의 '길이'와 '거리'를 살펴보겠습니다.

- 다음과 같은 성분 표시의 2차원 벡터가 있다고 가정합시다.

$$\mathbf{a} = (a_1, a_2)$$

- 이때 벡터 $\mathbf{a}$의 길이를 구해 보겠습니다.



- 그림 3-8을 살펴봅시다. 벡터의 길이를 $|\mathbf{a}|$이라고 할 때 피타고라스의 정리[1]에 의해 다음과 같은 식이 성립합니다.

$$|\mathbf{a}|^2 = a_1^2 + a_2^2$$

[1] 직각심각형의 빗변의 제곱은 나머지 두 변의 제곱을 더한 것과 같다는 정리입니다

- 이때 양변에 제곱근을 구하면 다음과 같이 표현할 수 있습니다.

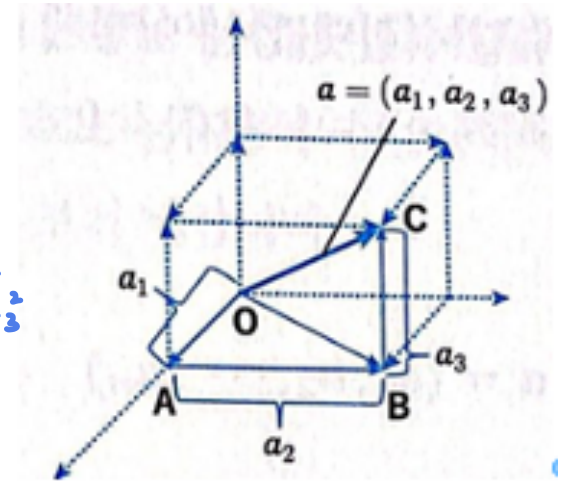
$$|\mathbf{a}| = \sqrt{a_1^2 + a_2^2}$$

- 이 식이 성분 표시 방법을 사용한 2차원 벡터의 길이 공식입니다. 이때의 벡터 길이를 벡터의 절댓값이라고도 합니다.
- 그러면 3차원 벡터의 길이(절댓값)는 어떻게 구할까요? 이해를 돕기 위해 그림 3-9를 살펴봅시다.

$|a| = \pm\sqrt{a_1^2+a_2^2+a_3^2}$

>0

$\Rightarrow |a| = \sqrt{a_1^2+a_2^2+a_3^2}$



$|(a_1, a_2)| = \sqrt{a_1^2+a_2^2}$

$\left(\sqrt{a_1^2+a_2^2}\right)^2 + a_3^2 = |a|^2$

$= a_1^2 + a_2^2 + a_3^2$

- 이 그림에서 다음과 같은 벡터 길이(절댓값) OC를 구해 봅시다.

$$\mathbf{a} = \overrightarrow{OC} = (a_1, a_2, a_3)$$

- 삼각형 OAB와 삼각형 OBC는 둘 다 직각삼각형입니다. 두 삼각형에 피타고라스의 정리를 적용하면 다음과 같습니다.

$$OA^2 + AB^2 = OB^2, \quad OB^2 + BC^2 = OC^2$$

- 이때 앞의 식의 OB^2을 뒤의 식에 대입하면 다음과 같이 정리할 수 있습니다.

$$OA^2 + AB^2 + BC^2 = OC^2$$

- $OA = a_1, AB = a_2, BC = a_3$로 다음과 같이 바꿔 쓸 수 있습니다.

$$OC^2 = a_1^2 + a_2^2 + a_3^2$$

- 이때 양변에 제곱근을 구하면 다음과 같이 표현할 수 있습니다.

$$OC = \sqrt{a_1^2 + a_2^2 + a_3^2}$$

- 이 식을 벡터의 길이(절댓값)로 다시 표현하면 다음과 같은 모양이 됩니다.

$$|\mathbf{a}| = \sqrt{a_1^2 + a_2^2 + a_3^2}$$

이것이 3차원 벡터의 길이(절댓값) 공식입니다.

- 그렇다면 이 식을 일반화한 n차원 벡터의 길이(절댓값)는 어떻게 구할까요? 아쉽게도 이 벡터는 그림으로 표현하지 못합니다. n차원의 벡터 길이가 얼마나 될지는 모르겠지만 2차원과 3차원 벡터의 길이를 구했던 방법을 확장해 봅시다.

$$\mathbf{a} = (a_1, a_2, \cdots, a_n)$$

- 벡터 $\mathbf{a}$가 위와 같을 때 길이 $|\mathbf{a}|$는 다음과 같은 방법으로 구할 수 있을 거라 짐작해 볼 수 있습니다.

$$|\mathbf{a}| = \sqrt{a_1^2 + a_2^2 + \cdots + a_n^2}$$

- 이 식이 n차원 벡터의 절댓값 공식입니다.

- 다음은 $\sum$ 기호에 대해 알아봅시다. $\sum$를 사용하면 위와 같은 식에서 여러 항목을 더해야 할 때 '...' 같은 생략 표현을 쓰지 않아도 됩니다.
- 위의 식에서 근호 안에 있는 부분은 다음과 같습니다.

$$a_1^2 + a_2^2 + \cdots + a_n^2$$

  이 식은 'k의 값을 1에서 n까지 변화시켰을 때 값을 모두 더한 것'입니다.
- 이를 $\sum$를 사용해 식으로 표현하면 다음과 같습니다.

$$\sum_{k=1}^{n} a_k^2$$

- 이어서 n차원 벡터 $\mathbf{a}$의 절댓값인 $|\mathbf{a}|$의 공식을 $\sum$를 써서 다시 표현하면 다음과 같이 쓸 수 있습니다.

$$|\mathbf{a}| = \sqrt{\sum_{k=1}^{n} a_k^2}$$

- $\sum$가 포함된 식은 읽기 쉽지 않지만 머신러닝에서는 피할 수 없는 표현이기 때문에 조금씩 친해질 필요가 있습니다. 단 앞에서 본 $\sum$가 없는 전개식이 직관적으로 이해하기 쉬우므로 $\sum$가 익숙해지기 전까지는 덧셈으로 전개된 식을 머릿속에 떠올려 보기 바랍니다.

- 다음은 두 벡터 $\mathbf{a}$와 $\mathbf{b}$의 거리에 대해 생각해 봅시다. 이번 절의 내용은 지금까지 나온 것을 모두 정리하는 것이기도 합니다. 결론부터 말하자면 벡터 $\mathbf{a}$와 $\mathbf{b}$의 뺄셈에다 절댓값을 적용하면 벡터 간의 거리가 나옵니다.
- 예를 들어, 두 개의 2차원 벡터를 성분으로 표시해 봅시다.

$$\mathbf{a} = (a_1, a_2),\ \mathbf{b} = (b_1, b_2)$$

- 이때 벡터 $\mathbf{a}$와 $\mathbf{b}$의 거리 d는 다음과 같은 식으로 표현할 수 있습니다.

$$d = |\mathbf{a} - \mathbf{b}| = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$

- 다음과 같이 3차원 벡터로 확장해 봅시다.

$$\mathbf{a} = (a_1, a_2, a_3),\ \mathbf{b} = (b_1, b_2, b_3)$$

- 이때 벡터 $\mathbf{a}$와 $\mathbf{b}$의 거리 d는 다음과 같은 식으로 표현할 수 있습니다.

$$d = |\mathbf{a} - \mathbf{b}| = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + (a_3 - b_3)^2}$$

- 이어서 n차원으로도 확장해 봅시다.

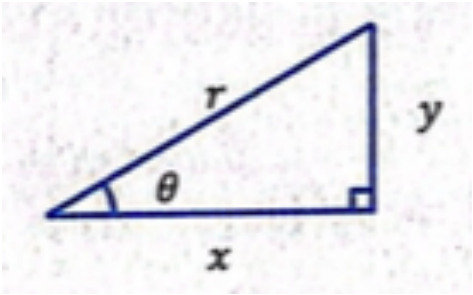
$$\mathbf{a} = (a_1, a_2, \cdots, a_n),\ \mathbf{b} = (b_1, b_2, \cdots, b_n)$$

- 이제는 두 개의 벡터 간의 거리 d를 다음과 같은 식으로 표현할 수 있다는 것을 어렵지 않게 짐작할 수 있을 것입니다.

$$\begin{aligned} d &= |\mathbf{a} - \mathbf{b}| \\ &= \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_n - b_n)^2} \\ &= \sqrt{\sum_{k=1}^{n} (a_k - b_k)^2} \end{aligned}$$

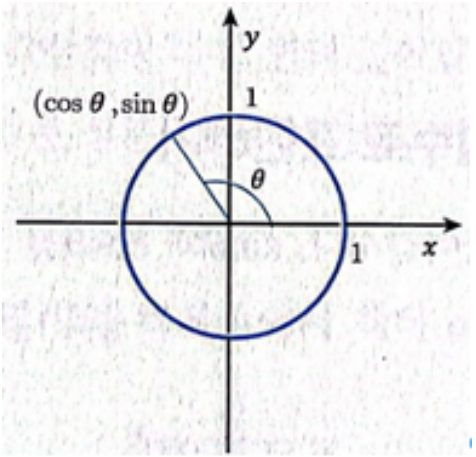
이번 절에서 삼각함수가 갑자기 나오는 이유는 3.5절에 나올 내적과 밀접한 관계가 있기 때문입니다. 여기서는 내적과의 관계를 염두에 두면서 삼각함수에 대해 알아봅시다.

- 우선 다른 수학 참고서처럼 삼각비의 정의부터 시작합니다.

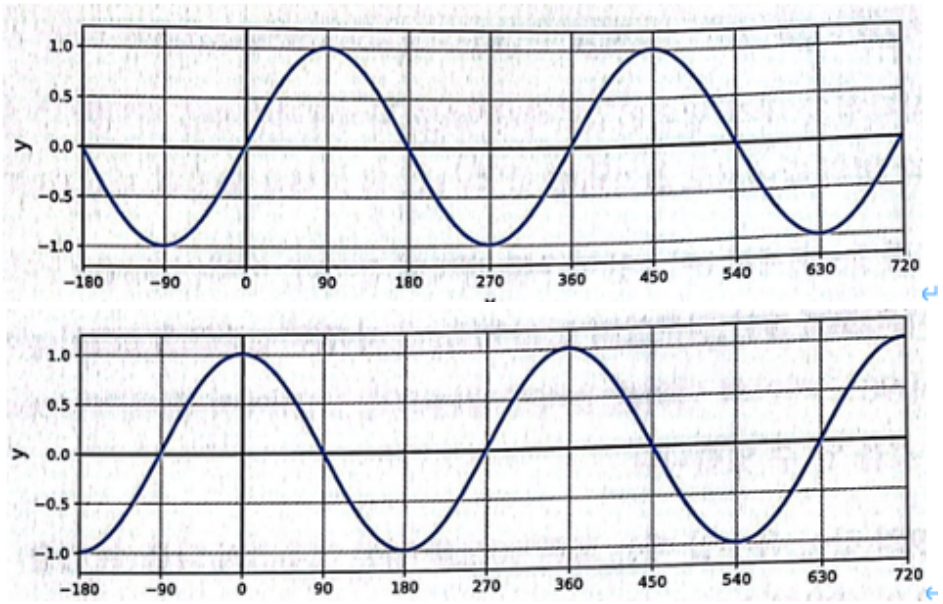


- 그림 3-10을 살펴봅시다. 직각삼각형은 그림의 내각 θ값을 고정했을 때 변의 길이가 달라지더라도 삼각형의 모양 자체는 큰 변화가 없습니다. 그래서 같은 변 간의 비율은 항상 일정하게 유지됩니다. 이때의 비율을 '삼각비(trigonometry ratio)'라 하고 이 값은 각도 θ에 의해 달라집니다. 삼각비는 다음과 같은 식으로 표현합니다.

- 삼각비에서는 직각삼각형의 내각 범위인 0도에서 90도 사이에서만 θ를 정의할 수 있었습니다. 이번에는 직각삼각형의 내각 범위 밖에서도 삼각비를 알 수 있도록 개념을 확장해 보겠습니다.
- 이해를 돕기 위해 그림 3-10에서 r이 1인 경우를 생각해 봅시다. 이때는 $\cos\theta = x$이고 $\sin\theta = y$인 상태입니다. 이번에는 그림 3-11에서 반지름은 1이고, 사분면의 원점을 중심으로 하는 원이 있다고 생각해 봅시다.
- x축이 양인 방향에서 각도가 θ인 원주상의 점이 있을 때 이 점의 x좌표를 $\cos\theta$, y좌표를 $\sin\theta$라고 합시다. θ값이 0도에서 90도 사이의 범위를 움직일 때는 그림 3-10에서 본 것처럼 $\cos\theta$와 $\sin\theta$의 삼각비가 똑같이 적용되는 것을 알 수 있습니다.
- 확장된 정의에서는 θ의 값이 음수일 때는 물론 90도를 넘는 경우까지 다룰 수 있습니다. 이렇게 삼각비의 개념이 확장된 것을 삼각함수라고 합니다.



- 앞에서 삼각함수의 정의를 살펴봤는데 이번에는 x축을 각도 θ, y축을 실수로 하는 삼각함수의 그래프를 그려봅시다.
- 그래프는 그림 3-12와 그림 3-13과 같이 깔끔한 파형으로 그려집니다. 그림 3-12의 곡선은 정현곡선(正弦曲線) 또는 사인 곡선 (sign curve)이라 하고 그림 3-13의 곡선은 여현곡선(餘弦曲線) 또는 코사인 곡선 (cosine curve)이라고 합니다.
- 두 그림을 살펴보면 $\sin\theta$ 함수를 x축의 음의 방향으로 90도 만큼 평행 이동시켰을 때 $\cos\theta$ 함수가 되는 것을 알 수 있습니다.



- 다시 그림 3-10과 삼각비의 정의로 돌아갑시다. $\sin\theta$와 $\cos\theta$를 정의한 식에서 양변에 r을 곱하면 다음과 같은 식이 됩니다.

$$x = r\cos\theta$$
$$y = r\sin\theta$$

- 이 식은 뒤에 나올 내적에서 중요한 의미를 가지므로 꼭 기억해 두기 바랍니다.[2]

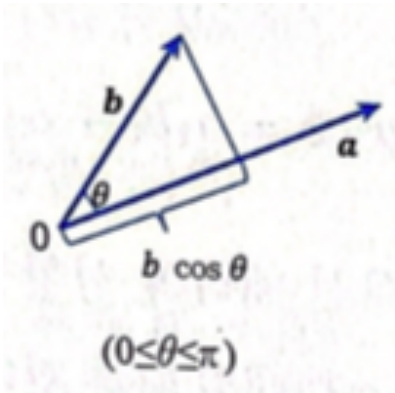
[2] 극좌표계(極座標系)와 직교좌표계(直交座標系)를 변환하는 식입니다.

드디어 내적에 대해 살펴볼 차례입니다. 3.2.3절에서는 '스칼라배'라고 하는 스칼라와 벡터의 곱셈에 대해 알아봤습니다. 이번에 다룰 내용은 또 다른 곱셈 연산으로 벡터끼리 곱하는 '내적'입니다. 고등학교 수학에서도 쉽게 이해하기 어려운 개념이지만 언제나 그랬듯이 우선은 2차원 벡터를 예로 들어 그림으로 설명해 보겠습니다.

- 벡터 $\mathbf{a}$와 $\mathbf{b}$가 이루는 각도를 θ라고 할 때 다음 식을 2차원 벡터 $\mathbf{a}$와 $\mathbf{b}$ 간의 내적으로 정의합니다.

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}| \cos\theta$$

- 이 식만 보면 내적이 수학적으로 어떤 의미가 있는지 감을 잡기 어렵습니다. 그래서 그림 3-14를 살펴 보겠습니다.



- 먼저 벡터 $\mathbf{b}$의 끝점에서 벡터 $\mathbf{a}$가 수직이 되도록 수선을 긋습니다.
- $|\mathbf{b}|\cos\theta$는 앞 절에서 살펴본 것처럼 직각삼각형의 한 변의 길이입니다. 이를 다르게 설명하면 벡터 $\mathbf{b}$를 벡터 $\mathbf{a}$와 같은 방향의 성분과 그렇지 않은 성분으로 분해한다고 할 때 $|\mathbf{b}|\cos\theta$의 길이는 벡터 $\mathbf{a}$와 같은 방향의 성분의 길이와 같습니다.
- 결국 벡터 $\mathbf{a}$와 $\mathbf{b}$의 내적은 벡터 $\mathbf{a}$의 길이와 벡터 $\mathbf{b}$에서 벡터 $\mathbf{a}$와 같은 방향의 성분 길이를 곱한 것이라고 할 수 있습니다.
- 내적의 성질을 조금 다른 관점에서 살펴봅시다. 벡터 $\mathbf{b}$의 길이 $|\mathbf{b}|$를 고정한 다음, θ값을 변경했을 때 내적 값에는 어떤 변화가 있을지 생각해 봅시다. 이때 θ값을 변경한다는 의미는 그림 3-14에서 두 벡터의 시작점이 중심이고 반지름이 $|\mathbf{b}|$인 원 주위를 돈다고 생각하면 됩니다.

- 그러면 표 3-1과 같은 결과가 나오는 것을 알 수 있습니다.

표 3-1 각도 θ와 내적 값의 관계

θ의 값	벡터 $\mathbf{a}$와 $\mathbf{b}$의 관계	내적 값
0도	두 벡터의 방향이 완전히 일치한 상태	최댓값
90도	두 벡터가 직교인 상태	0
180도	두 벡터의 방향이 완전히 반대인 상태	최솟값

- 이런 특징은 내적에서 상당히 중요한 성질이므로 잘 기억해 두기 바랍니다.

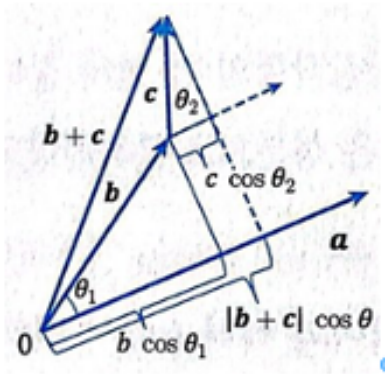
- 앞 절에서는 두 벡터의 절댓값과 이루는 각으로 내적을 정의했는데 이번에는 벡터의 성분으로 표현해 봅시다. 우선 벡터 $\mathbf{a}$와 $\mathbf{b}$의 성분이 다음과 같다고 가정합시다.

$$\mathbf{a} = (a_1, a_2),\ \mathbf{b} = (b_1, b_2)$$

- 이때 다음과 같은 식이 성립합니다

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2$$

- 이 식이 어떻게 성립하는지 알아봅시다. 우선 내적의 선형성이 성립한다는 것을 직관적으로 이해합시다. 내적의 선형성이란 임의의 벡터 $\mathbf{a}, \mathbf{b}, \mathbf{c}$에 대해 다음 관계가 성립하는 것을 의미합니다.[3]



[3]내적의 선형성은 벡터 $\mathbf{a}, \mathbf{b}, \mathbf{c}$가 있을 때 벡터 $\mathbf{a}, \mathbf{b}$와 스칼라 $\mathbf{c}$가 있을 때 $(c\mathbf{a}) \cdot \mathbf{b} = c(\mathbf{a} \cdot \mathbf{b})$와 $\mathbf{a} \cdot (c\mathbf{b}) = c(\mathbf{a} \cdot \mathbf{b})$도 성립합니다. 여기서는 그림 3-15로 설명하기 위해 $\mathbf{a} \cdot (\mathbf{b} + \mathbf{c}) = \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \cdot \mathbf{c}$만 언급했습니다.

- 그림 3-15를 살펴봅시다. 이 식이 성립한다는 말은 앞에서 설명한 내용으로부터 다음 식이 성립한다는 것을 의미합니다.

$$\begin{aligned}&(\mathbf{b}+\mathbf{c} \text{ 벡터에서 벡터 } \mathbf{a}\text{와 같은 방향의 성분})\\ =\ &(\text{벡터 } \mathbf{b}\text{에서 벡터 } \mathbf{a}\text{와 같은 방향의 성분})\\ +\ &(\text{벡터 } \mathbf{c}\text{에서 벡터 } \mathbf{a}\text{와 같은 방향의 성분})\end{aligned}$$

  이 식의 의미는 그림 3-15를 보면 쉽게 이해할 수 있습니다.
- 만약 선형성이 옳다고 한다면 두 벡터를 다음과 같이 성분으로 표시할 때

$$\mathbf{a} = (a_1, a_2),\ \mathbf{b} = (b_1, b_2)$$

  다음과 같은 벡터로 좀 더 작게 분해할 수 있습니다.

$$\mathbf{a}_1 = (a_1, 0),\ \mathbf{a}_2 = (0, a_2),\ \mathbf{b}_1 = (b_1, 0),\ \mathbf{b}_2 = (0, b_2)$$

- 이때 벡터 $\mathbf{a}$와 $\mathbf{b}$는 다음과 같이 표현할 수 있습니다.

$$\mathbf{a} = \mathbf{a}_1 + \mathbf{a}_2,\ \mathbf{b} = \mathbf{b}_1 + \mathbf{b}_2$$

- 벡터 $\mathbf{a}$와 $\mathbf{b}$를 곱하면 다음과 같은 식이 됩니다.

$$\mathbf{a} \cdot \mathbf{b} = (\mathbf{a}_1 + \mathbf{a}_2) \cdot (\mathbf{b}_1 + \mathbf{b}_2)$$

- 여기서 선형성의 특징을 활용하면 다음과 같이 식을 전개할 수 있습니다.

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{a}_1 \cdot \mathbf{b}_1 + \mathbf{a}_1 \cdot \mathbf{b}_2 + \mathbf{a}_2 \cdot \mathbf{b}_1 + \mathbf{a}_2 \cdot \mathbf{b}_2$$

  이때 $\mathbf{a}_1 \cdot \mathbf{b}_2$와 $\mathbf{a}_2 \cdot \mathbf{b}_1$는 두 벡터가 서로 직교하고 있기 때문에 값은 0이 됩니다. 또한 $\mathbf{a}_1 \cdot \mathbf{b}_1$는 두 벡터의 방향이 같기 때문에 $a_1 b_1$이 되는 것을 알 수 있습니다. $\mathbf{a}_2 \cdot \mathbf{b}_2$도 같은 이유로 $a_2 b_2$가 됩니다.
- 이러한 내용을 정리해 보면 다음 식이 성립하는 것을 알 수 있습니다.

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2$$

- 이것이 2차원 벡터의 내적을 벡터의 성분으로 정리한 공식입니다.

- 이제 3차원 벡터에서는 어떻게 나오는지 살펴봅시다. 다음과 같은 두 벡터 $\mathbf{a}, \mathbf{b}$가 있다고 가정합시다.

$$\mathbf{a} = (a_1, a_2, a_3),\ \mathbf{b} = (b_1, b_2, b_3)$$

  앞서 2차원 벡터에서 본 내적의 선형성은 3차원 벡터에서도 성립할 것 같습니다. 그래서 2차원 벡터 때와 마찬가지로 다음과 같이 벡터를 분해해 봤습니다.

$$\mathbf{a}_1 = (a_1, 0, 0),\ \mathbf{a}_2 = (0, a_2, 0),\ \mathbf{a}_3 = (0, 0, a_3)$$
$$\mathbf{b}_1 = (b_1, 0, 0),\ \mathbf{b}_2 = (0, b_2, 0),\ \ \mathbf{b}_3 = (0, 0, b_3)$$

- 이때 벡터 $\mathbf{a}$와 $\mathbf{b}$는 다음과 같이 표현할 수 있습니다.

$$\mathbf{a} = \mathbf{a}_1 + \mathbf{a}_2 + \mathbf{a}_3,\ \mathbf{b} = \mathbf{b}_1 + \mathbf{b}_2 + \mathbf{b}_3$$

- 여기서 2차원 벡터의 경우와 마찬가지로 선형성의 특징을 활용해 식을 전개한 다음, 직교하는 벡터는 0으로, 같은 방향인 벡터는 성분으로 표시하면 다음과 같이 식을 정리할 수 있습니다.

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + a_3 b_3$$

- 여기까지 해봤다면 n차원으로의 확장은 쉽습니다. 다음과 같은 벡터가 있다고 할 때

$$\mathbf{a} = (a_1, a_2, \cdots, a_n),\ \mathbf{b} = (b_1, b_2, \cdots, b_n)$$

- 지금까지의 내용을 일반화해 보면 다음과 같은 식이 성립하는 것을 알 수 있습니다

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n = \sum_{k=1}^{n} a_k b_k$$

- 이것이 n차원 벡터의 내적을 벡터의 성분으로 정리한 공식입니다.

- 다음과 같은 성분의 2차원 벡터가 있다고 가정합시다.

$$\mathbf{a} = (a_1, a_2),\ \mathbf{b} = (b_1, b_2)$$

- 두 벡터 사이의 각 θ를 구하는 문제가 있다면 앞 절에서 도출했던 다음 식으로 어렵지 않게 풀 수 있습니다.

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}| \cos\theta = a_1 b_1 + a_2 b_2$$

- 이 식을 $\cos\theta$를 기준으로 다시 쓰면 다음과 같습니다. 중간 식의 분모에 있는 벡터의 절댓값은 3.3.1절에서 배운 방법을 이용해 마지막 식에서 벡터의 성분으로 풀어서 썼습니다.

$$\cos\theta = \frac{a_1 b_1 + a_2 b_2}{|\mathbf{a}||\mathbf{b}|} = \frac{a_1 b_1 + a_2 b_2}{\sqrt{a_1^2 + a_2^2}\sqrt{b_1^2 + b_2^2}}$$

- 이때 $\cos\theta$의 θ를 구하려면 $\arccos(x)$ 함수를 사용하면 됩니다. 결국 2차원 벡터의 성분을 알면 두 벡터가 이루는 각을 구할 수 있다는 말입니다.

- 2차원 벡터와 이루는 각: 비슷한 방법으로 3차원 벡터가 이루는 각도 다음 식과 같이 구할 수 있습니다.

$$\cos\theta = \frac{a_1 b_1 + a_2 b_2 + a_3 b_3}{\sqrt{a_1^2 + a_2^2 + a_3^2}\sqrt{b_1^2 + b_2^2 + b_3^2}}$$

- 이번에는 앞에서 살펴본 공식을 n차원으로 확장해 봅시다. 확장한 공식은 다음과 같습니다.

$$\cos\theta = \frac{a_1 b_1 + a_2 b_2 + \cdots + a_n b_n}{\sqrt{a_1^2 + a_2^2 + \cdots + a_n^2}\sqrt{b_1^2 + b_2^2 + \cdots + b_n^2}} = \frac{\sum_{k=1}^{n} a_k b_k}{\sqrt{\sum_{k=1}^{n} a_k^2}\sqrt{\sum_{k=1}^{n} b_k^2}}$$

- 형식적으로는 이런 식이 나왔지만 정작 궁금한 것은 '4차원 이상의 두 벡터가 이루는 각은 과연 어떤 것일까?라는 점입니다. 사실 4차원부터는 벡터가 실제로 어떻게 생겼는지 상상조차 하기 어렵기 때문에 애당초 각도가 어떨지 짐작하기 어렵습니다.
- 다만 차원이 100차원이라 하더라도 이 식을 통해 코사인과 비슷한 결과가 나오는 것은 확실합니다. 그리고 최소한 이 식에서 구한 값이 1에 가까울수록 '두 벡터가 이룬 각은 작다, 즉 '두 벡터의 방향은 비슷하다'라고 할 수 있습니다.
- 한편 이렇게 다차원 벡터에서 위와 같은 수식을 계산한 값을 '코사인 유사도(cosine similarity)'라고 합니다. 코사인 유사도는 서로 다른 벡터가 얼마나 비슷한 방향을 가리키고 있는지 가늠하는 지표로 자주 활용됩니다.

칼럼: 코사인 유사도의 활용

- 서로 다른 n차원 벡터가 얼마나 가깝게 위치하는지 알아내는 방법은 인공지능 분야에서 자주 논의되는 주제 중 하나입니다. 이때 활용하는 것이 앞 절에서 언급한 '코사인 유사도'인데 이해를 돕기 위해 두 가지 예를 들어 봅시다.
- 첫 번째 예는 'Word2Vec'의 응용입니다. Word2Vec은 최근 주목받고 있는 텍스트 분석 방법에서 '가까이에 위치한 단어는 서로 연관성이 있다'는 전제로 대량의 텍스트 데이터를 학습시킨 다음, 어떤 단어와 100차원 정도의 수치 벡터 간의 대응표를 만드는 기법입니다.
- 그 결과로 만들어진 수치 벡터는 아주 재미있는 성질을 가지고 있는데, 다음과 같은 연산이 가능하다고 합니다.

('왕'을 표현하는 수치 벡터) − ('여왕'을 표현하는 수치 벡터)
= ('남자'를 표현하는 수치 벡터) − ('여자'를 표현하는 수치 벡터)

- 중요한 것은 언어를 구성하는 주요 단어가 숫자 벡터로 표현이 된다는 점입니다. 이러한 수치 벡터를 입력하면 '어떤 단어와 유사한 단어'를 찾을 수 있다는 말이며, 이때 활용되는 알고리즘이 바로 코사인 유사도입니다.

칼럼: 코사인 유사도의 활용

- 두 번째 예로는 IBM의 인공지능 API인 'Personality Insights'입니다. 이 API는 어떤 사람이 트위터 등에 쓴 글을 입력받아 'Big5 성격 테스트'[4]와 비슷한 점검 결과를 출력하는데 이를 통해 글쓴이의 특징을 가늠할 수 있다고 합니다.
- 결과는 5차원의수치 벡터로 나오는데 사람과 사람 간의 상성(相性)을 코사인 유사도로 평가할 수 있다는 말입니다.
- 자기 자신과 다른 사람의 궁합을 확인하고 싶다면 두 사람의 Personality Insights 결과를 받은 다음, 코사인 유사도를 계산해 보는 것도 재미 있을 것 같습니다.

[4] 1980년대 심리학자 폴 코스타와 로버트 맥레는 사람의 성격 특성을 다섯 가지 (신경성, 외향성, 친화성, 성실성, 경험에 의한 개방성)로 분류했는데 자신이 속한 문화권과 상관없이 그 사람의 기질을 잘 나타낸다고 알려져 있습니다.

이번 절에서는 벡터의 개념을 확장한 '행렬'에 대해 알아보고 행렬과 벡터 간의 곱셈도 해 보겠습니다. 수학에서는 '선형대수'에서 해당하는 내용인데 여기에는 '역행렬'이나 '고윳값, '고유 벡터'와 같은 중요한 개념과 공식, 정리 등이 나옵니다. 다만 이 책의 목표가 최단 코스로 딥러닝 알고리즘을 배우는 것이므로 딥러닝에 필요한 최소한의 내용만 살펴 보겠습니다.

- 그림 3-16을 살펴봅시다.

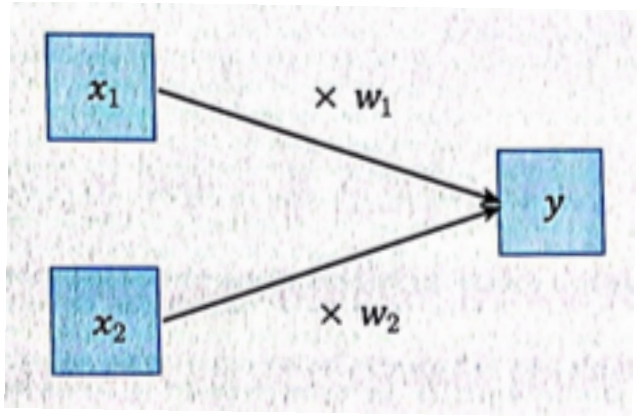


그림 3-16 입력 노드 2개, 출력 노드 1개의 네트워크

- 이 식은 x_1과 x_2를 입력받는 머신러닝 모델을 간단히 표현한 것으로 출력 변수 y는 다음과 같이 표현할 수 있습니다.

$$y = w_1 x_1 + w_2 x_2 \tag{1}$$

- 이 식은 입력 노드에 계수를 곱하고 그 결과를 모두 더해 하나의 출력으로 만들고 있습니다. 머신러닝에서는 이런 식의 결과(출력)를 다음 노드의 입력으로 다시 넣는 방식이 자주 사용됩니다.
- 수식 (1)의 우변을 살펴 보면 두 개의 벡터 $\mathbf{w} = (w_1, w_2)$와 $\mathbf{x} = (x_1, x_2)$의 내적으로 볼 수 있는데 이식을 고쳐 쓰면 다음과 같이 표현할 수 있습니다.

$$y = \mathbf{w} \cdot \mathbf{x} \tag{2}$$

- 벡터를 사용한 표현 방식은 수식을 짧고 간결하게 쓸 수 있다는 장점이 있습니다. 또한 파이썬에는 벡터를 연산하는 다양한 함수가 제공되기 때문에 프로그램 개발도 손쉽게 할 수 있습니다. 파이썬으로 어떻게 구현하는지에 대해서는 7장 이후의 실습편에서 다룰 예정입니다.

- 그림 3-17을 살펴봅시다. 앞서 살펴본 그림 3-16에서는 입력 노드가 2개, 출력 노드가 1개였지만 이번에는 출력 노드가 3개입니다. 이런 구조는 9장에서 소개할 '다중 클래스 분류'에서 사용합니다.

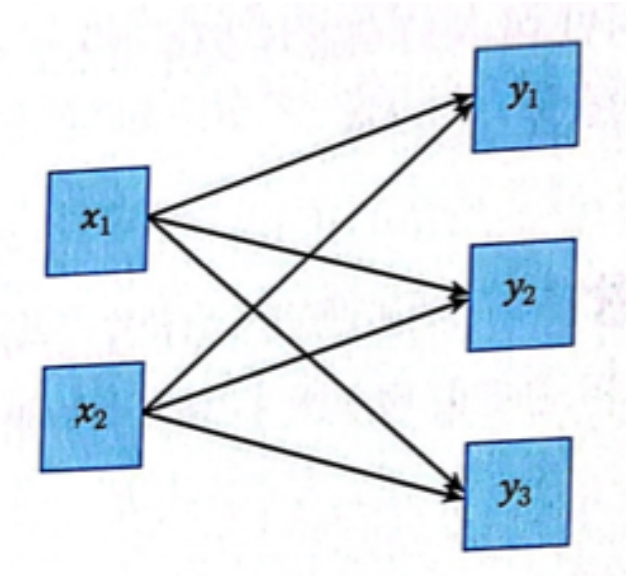


그림 3-17 입력 노드 2개, 출력 노드 3개로 구성된 네트워크

- 이때 가중치를 나타내는 매개변수는 $2 \times 3 = 6$이므로 총 6개가 필요한데 매개변수를 구분하는 첨자 번호를 1차원 요소처럼 쓰게 되면 어떤 것이 어떤 매개변수인지 알아보기 어려울 수 있습니다. 그래서 가중치 w의 첨자를 2차원으로 표현하는 방식이 사용됩니다.

- 예를 들어, 수식 (1)은 다음과 같이 표현할 수 있습니다.

$$\begin{aligned} y_1 &= w_{11}x_1 + w_{12}x_2 \\ y_2 &= w_{21}x_1 + w_{22}x_2 \\ y_3 &= w_{31}x_1 + w_{32}x_2 \end{aligned} \tag{3}$$

- 이 식의 w처럼 요소의 첨자가 2차원이고 가로, 세로로 데이터가 펼쳐진 데이터 표현 방식을 행렬이라고 합니다.[5]
- 벡터와 마찬가지로 행렬을 성분으로 표시해 보면 다음과 같은 모양이 됩니다.[6]

$$\mathbf{W} = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{pmatrix}$$

[5] 반면 1차원으로 데이터가 펼쳐진 것은 벡터입니다.
[6] 행렬 전체를 변수로 취급할 때는 대문자로 굵게 표현합니다.

- 한편 행렬의 정의에 따라 행렬과 벡터의 곱셈을 정의할 수 있습니다. 예를 들어, 다음과 같은 벡터 $\mathbf{x}$가 있다고 가정합시다.

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

- 이 벡터 $\mathbf{x}$와 행렬 $\mathbf{W}$ 간의 곱셈은 다음과 같이 정의할 수 있습니다.

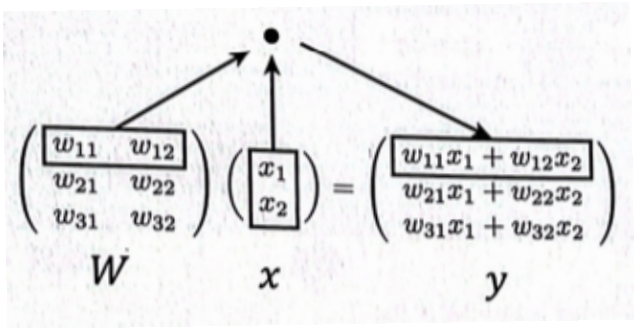


그림 3-18 행렬과 벡터의 곱셈

- 계산하는 요령으로 왼쪽 행렬의 가로 한 줄(행)과 오른쪽 벡터의 세로 한 줄(열)에 대한 내적을 구하면 되는데 이 과정을 모든 행과 열에 대해 반복하면 됩니다. 이것이 행렬과 벡터의 곱셈입니다.
- 이 예에서는 행렬 전체를 변수 $\mathbf{W}$로, 벡터를 $\mathbf{x}$로 표현하고 있습니다. 같은 방식으로 앞의 수식 (3)의 출력 벡터 (y_1, y_2, y_3)를 $\mathbf{y}$로 표현하면 다음과 같이 다시 쓸 수 있습니다.

$$\mathbf{y} = \mathbf{W}\mathbf{x} \tag{4}$$

- 파이썬에서는 수식 (4)와 같이 행렬과 벡터를 곱하는 식도 간단하게 구현할 수 있습니다. 구체적인 방법은 7장 이후의 실습편에서 자세히 살펴봅시다.

1. ML lec 03 - Linear Regression의 cost 최소화 알고리즘의 원리 설명 https://www.youtube.com/watch?v=TxIVr-nk1so&list=PLlMkM4tgfjnLSOjrEJN31gZATbcj_MpUm&index=7&t=1s
2. ML lab 03 - Linear Regression의 cost 최소화의 TensorFlow 구현 https://www.youtube.com/watch?v=YOEF9VqRuEA&list=PLlMkM4tgfjnLSOjrEJN31gZATbcj_MpUm&index=7
3. ML Lec 04 - multi-variable linear regression https://www.youtube.com/watch?v=kPxpJY6fRkY&list=PLlMkM4tgfjnLSOjrEJN31gZATbcj_MpUm&index=8
4. ML Lab 04-1- multi-variable linear regression을 TensorFlow에서 구현하기 https://www.youtube.com/watch?v=fZUV3xjoZSM&list=PLlMkM4tgfjnLSOjrEJN31gZATbcj_MpUm&index=9
5. ML Lab 04-2 - TensorFlow로 파일에서 데이타 읽어오기 https://www.youtube.com/watch?v=o2q4QNnoShY&list=PLlMkM4tgfjnLSOjrEJN31gZATbcj_MpUm&index=10