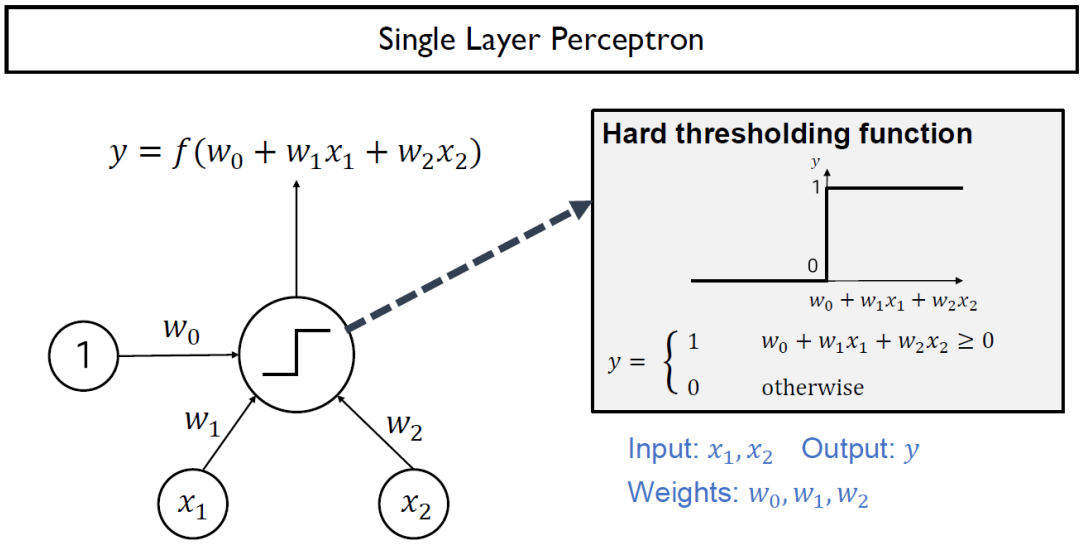
리드오프 2주차 전반부

1. 딥러닝과 경사하강법

지금까지 배운 이론들이 통계학 및 모델 학습에서 어떻게 응용되고, 어떤 의미를 가지는지 알아보기 위해 딥러닝에서의 핵심 개념인 역전파를 가볍게 다뤄보자.

퍼셉트론

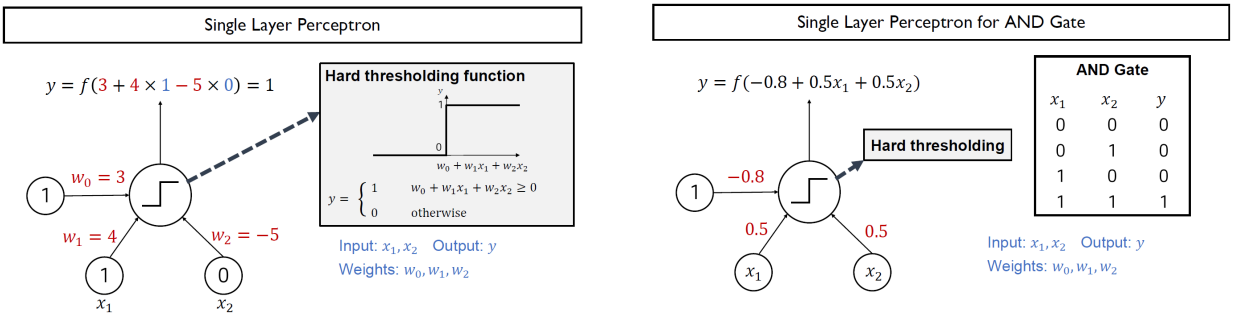
역전파를 이해하기 위해선 먼저 **퍼셉트론**(Perceptron)에 대한 이해가 필요하다. 퍼셉트론은 가장 고전적인 선형 분류 모델로, Support Vector Machine이나 딥러닝 등의 강력한 모델이 모두 퍼셉트론의 변형이다.



퍼셉트론이 작동하는 방식은 간단하다. 예시로 x_1과 x_2만이 입력되는 상황을 가정해보자.

퍼셉트론은 입력 정보 x_1과 x_2에 대해 각각 가중치 w_1과 w_2를 곱해 더한 후(가중합), 활성 함수(f)라는 것을 거쳐서 그 값을 다음 계층에 있는 뉴런(퍼셉트론)으로 전해주게 된다. 활성 함수는 가중 합의 값을 다음 계층의 뉴런으로 전달하기 위해 적절히 변형하는 함수임을 알아두자.

위 예시에서 활성 함수는 가중합의 값이 0보다 크면 1, 아니면 0을 반환하는 함수이고, 따라서 Output은 1 또는 0이 된다.



위는 퍼셉트론의 예시와, 이 간단한 예시를 이용해 AND라는 간단한 논리연산을 수행할 수 있는 퍼셉트론의 예시이다. 즉 이러한 퍼셉트론을 이용해 Classification의 기능을 수행할 수 있다는 것이다.

역전파

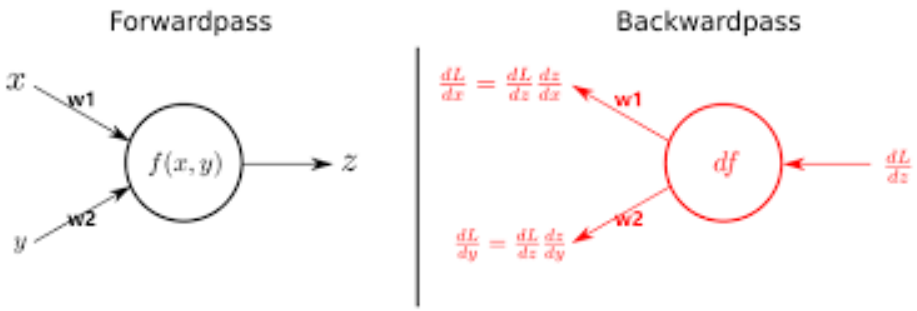
이제 역전파의 정의에 대해서 알아보자.

우리가 예측을 수행하고자 할 때 위 예시처럼 AND라는 기준(True Function)을 알고 있다면 퍼셉트론에서 우리가 직접 가중치를 설정하는 것이 가능할 것이다. 그러나 우리는 당연하게도, 데이터가 형성되는 True Function을 알고 있지 못하는 경우가 많다. 즉 우리는 가지고 있는 데이터를 기반으로 저 Weight(위 예시에서는 w_1과 w_2)와 Bias(위 예시에서는 w_0)를 조정해나가며 모델을 학습해야 한다는 것이다.

모델 학습은 손실 함수(Loss Function) 값을 최소화하는 파라미터를 구하는 과정이라고도 할 수 있다. 손실 함수는 지도학습 시 알고리즘이 예측한 값과 실제 정답의 차이를 비교하기 위한 함수이다. 회귀분석에서 실제 값과 예측 값 사이의 오차를 제곱해 합했던 $\sum_{i=1}^{n}(y_i - \beta_1 x_i)^2$ 역시 손실 함수의 일종이다.

즉 손실이 작아질수록 학습이 잘 이루어지고 있다고 해석할 수 있고, 손실 함수는 알고리즘을 평가하기 위한 지표라고도 볼 수 있다. 모델 학습은 손실 함수를 최소화하는 것이 목적이고, 이는 즉 최적화 문제이다.

역전파는 신경망의 각 노드가 가지고 있는 Weight와 Bias를 학습시키기 위한 알고리즘이다. 목표(Target)와 모델의 예측 결과(Output)가 얼마나 차이가 나는지 확인하고(손실 함수 계산) 그 오차를 바탕으로 손실 함수를 최소화하는 방향으로 가중치와 편향을 뒤에서부터 앞으로 갱신해가는 것을 의미한다. 역전파란 명칭도 바로 이처럼 뒤에서부터 다시 앞으로 거슬러 올라간다는 것에서 나온 것이고, 이제 우리는 손실 함수의 최소화(Optimization)를 위해 이전 시간에 배운 Gradient Descent 알고리즘을 사용하게 된다.



📌 **정리하자면**

목표 : 퍼셉트론, 또는 Neural Network가 예측을 잘 하는 것

그래서 최소화하고자 하는 것 : Target과 Prediction 사이의 오차인 Loss Function

최소화를 위해 조정하게 되는 Parameter : Weight와 Bias (w)

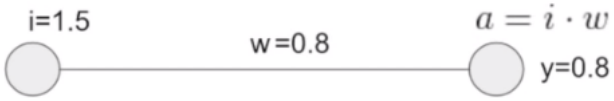
최적화 방법 : Gradient Descent

예제

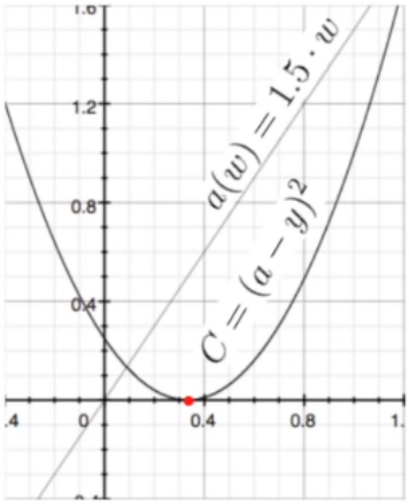
우선 매우 간단한 예제를 통해 역전파 알고리즘을 수행해보자.

$$C = (a-y)^2$$

$$a = i \cdot w$$



input (i)	desired output (y)	output (a)
1.5	0.5	1.2



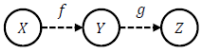
input으로 1.5가 주어졌고, 초기 weight가 0.8로, 단순히 i와 w를 곱해주기 때문에 출력된 결과는 1.2가 된다. 실제 y 값은 0.8이다. 손실 함수로는 MSE(Mean Square Error)를 사용하였으며 현재 인풋이 1개이므로 $(a-y)^2$ 이 손실 함수가 된다. 오른쪽에 보이는 그래프는 w에 따라 손실함수의 그래프를 나타낸 것이고, 최적의 w는 0.5/1.5 인 0.333…이 된다.

그렇다면 w가 초기 값인 0.8에서 0.333… 으로 조정되는 과정을 살펴보자.

손실 함수를 최소화하는 것이 목적이므로, 우선 $(a-y)^2$의 Gradient를 계산한다. a에 대해 편미분한 값인 $2(a-y)$가 될 것이다. 그러나 우리는 w를 조정하는 것이 목적이므로 저 값을 w에 대해 나타내주어야 하는데, 이 때 **Chain Rule**이 사용된다.

One Variable Case (Scalar Function)

- 이때까지는 독립변수가 종속변수에 직접적으로 영향을 미치는 경우를 살펴보았다. 그러나 X가 Y에 영향을 미치고, 다시 Y가 Z에 영향을 미치는 경우를 보자.



- 예를 들어, 코카콜라의 원자재 값이 올랐다고 하자. 이로 인해 코카콜라의 가격이 올랐고, 경쟁사인 펩시를 찾는 경우가 많아졌다. 이때 코카콜라 원자재 1원 만큼 올랐을 때, 펩시를 찾는 사람들은 얼마나 많아졌는지 구해보자.
- 원자재 비용 인상이 가격에 미치는 영향은 $\frac{df}{dx}$로 구할 수 있다. 마찬가지로 코카콜라의 가격이 펩시에 수요에 미치는 영향은 $\frac{dg}{dy}$를 통해 구할 수 있다. 즉, 코카콜라 비용 1원 이상은 $\frac{df}{dx}$원의 가격 인상을 가져오고, 1원의 가격 인상은 $\frac{dg}{dx}$명의 수요를 증가시킨다.

$$\frac{dZ}{dX} = \frac{df \circ g}{dX} = \frac{df}{dX} \times \frac{dg}{dY} = \frac{df}{dX} \times \frac{dg}{df}$$

1주차에서 배운 것을 떠올려보자. 다음 상황을 이 예시 상황에 적용시켜보면, w에 따라 a가 영향을 받고, a에 따라 $C = (a-y)^2$가 영향을 받는 상황이다.
그러므로
C의 w에 의한 변화율을 계산하기 위해서는 $\frac{\partial C}{\partial w}$를 계산해야 하고, 이는 Chain Rule에 의해

$$\frac{\partial C}{\partial w} = \frac{\partial C}{\partial a} \cdot \frac{\partial a}{\partial w}$$

로 계산되는 것이다.

$a = 1.5w$ 이므로 $\frac{\partial a}{\partial w}$ 는 1.5, $\frac{\partial C}{\partial a}$는 이전에 계산한 $2(a-y)$이다. 따라서

$$\frac{\partial C}{\partial w} = 3(a-y) = 4.5w - 1.5$$

이제 Gradient를 계산했으니 가중치를 갱신하면 된다. Gradient Descent의 공식을 다시 Remind해보면 다음의 형태이고, 이를 적용해보자

$$x_{i+1} = x_i - \alpha \frac{df}{dx}(x_i)$$

step size를 0.2로 둔다고 했을 때

$$w_1 = w_0 - \alpha \cdot \frac{\partial C}{\partial w} = 0.8 - 0.2(4.5 \cdot 0.8 - 1.5) = 0.38$$

$$w_2 = w_1 - \alpha \cdot \frac{\partial C}{\partial w} = 0.38 - 0.2(4.5 \cdot 0.38 - 1.5) = 0.338$$

와 같은 식으로 점점 w는 0.333…에 수렴하게 된다.

이것이 Gradient Descent에 기반한 역전파의 핵심 원리이며, 이는 초기값을 설정하고 나온 Output으로부터 역으로 거슬러 올라가며 가중치를 조정하는 것이다.

물론 위 예시에서는 $2(a-y)=2(1.5\cdot w\ -0.5)=0$ 의 방정식을 풀어 최적의 w를 찾을 수 있다. 이전에도 가볍게 설명했지만, 그렇다면 왜 굳이 딥러닝은 Gradient Descent 기반의 방식을 사용하는 것일까? 이는 아래의 Additional Topic에서 한번 더 이해해볼 수 있을 것이다.

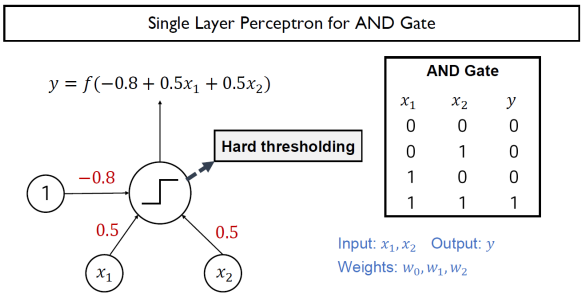
Additional Topic

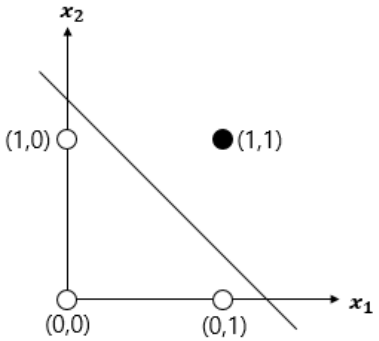
※ 과제 풀이의 이해를 돕기 위해 Additional Topic으로 다층 퍼셉트론과 비선형함수를 다룹니다. 추후 딥러닝(CV, NLP) 클린업에서도 큰 도움이 될 것이라고 생각합니다.

> 📌 **핵심부터 말하자면**
>
> 1. 퍼셉트론은 더 복잡한 문제를 해결하기 위해 층(Layer)을 쌓는다. 층이 많이 쌓인 신경망을 두고 **딥러닝**, 또는 심층신경망(**Deep Neural Network**)이라고 한다.
> 2. 선형 활성화 함수는 Layer가 쌓여도 **Layer가 하나인 것과 크게 다르지 않다**. 그러므로 활성화 함수로 비선형함수를 사용해야 한다.
> 3. 비선형 활성화 함수는 당연하게도, 미분을 하고 미분 값이 0이 되는 지점을 찾는 것이 힘들다. 그러므로 Gradient Descent와 같은 방식의 최적화가 더욱 중요해진다.

다층 퍼셉트론

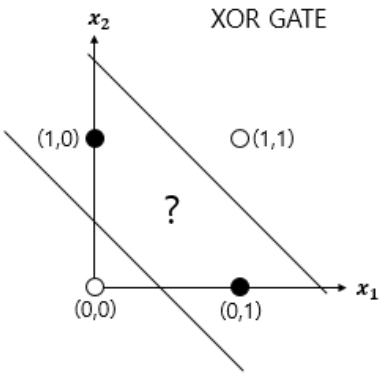
사실 지금까지 다룬 하나의 퍼셉트론으로는 매우 간단한 연산이나 문제만 해결할 수 있다.



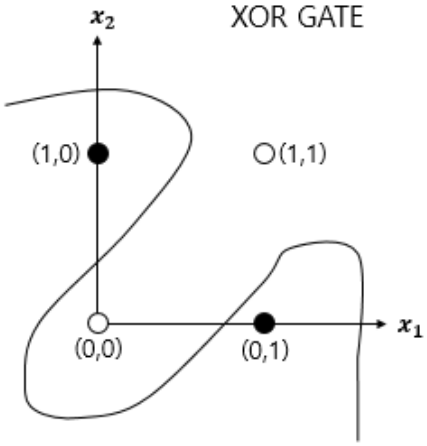


x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

맨 위에서 다루었던 AND 문제를 푸는 퍼셉트론을 시각화하면 오른쪽과 같고, 즉 직선 하나로 두 영역을 나눌 수 있는 문제에 대해 풀이하고 있는 것이기 때문이다. 이를 두고 단층 퍼셉트론은 선형 영역에 대해서만 분리가 가능하다고 한다.

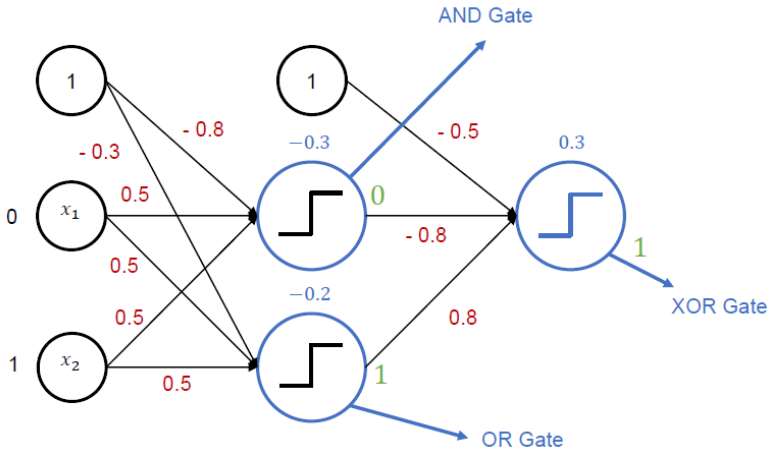


x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



그렇다면 XOR 문제는 어떨까? XOR 문제는 입력값 두 개가 서로 다른 값을 갖고 있을때에만 출력값이 1이 되고, 입력값 두 개가 서로 같은 값을 가지면 출력값이 0이 되는 연산이다. XOR 연산을 시각화해보면 다음과 같다.

하얀색 원과 검은색 원을 직선 하나로 나누는 것은 불가능하다. 즉, 단층 퍼셉트론으로는 XOR 연산을 구현하는 것이 불가능하다는 것. 그러나 XOR 연산은 직선이 아닌 곡선. 비선형 영역으로 분리함으로써 구현이 가능하다.



XOR 게이트는 기존의 AND, OR 게이트를 조합하여 만들 수 있다. 가중치를 빨간 색과 같이 설정하여 $x1$과 $x2$가 각각 (1,1) (1,0)일 때를 직접 대입해보자.

이처럼 굉장히 간단한 문제만을 수행할 수 있었던 하나의 퍼셉트론을 여러 단계에 걸쳐 구성했을 때 더 복잡한 문제도 풀어낼 수 있는 능력을 보이게 된다.

다층 퍼셉트론과 단층 퍼셉트론의 차이는 단층 퍼셉트론은 입력층과 출력층만이 존재하지만, 다층 퍼셉트론은 중간에 층을 더 추가하였다는 점이다. 이렇게 입력층과 출력층 사이에 존재하는 층을 **은닉층(hidden layer)**이라고 한다. 즉, 다층 퍼셉트론은 중간에 은닉층이 존재한다는 점이 단층 퍼셉트론과 다르다.

'비선형' 활성화 함수

활성화 함수(Activation function)는 입력을 받아 수학적 변환을 수행하고 출력을 생성하여 다음 계층으로 전달해주기 위한 함수이다. 앞서 0보다 크면 1을, 0보다 작으면 0을 출력하는 함수도 활성화 함수의 일종이다.

활성화 함수의 특징은 **선형 함수가 아닌 비선형 함수**여야 한다는 점인데, 활성화 함수로 선형 함수를 사용하게 되면 은닉층을 쌓을 수가 없기 때문이다.

예를 들어 활성화 함수로 $f(x) = Wx$ 라는 선형 함수를 선택하고, 층을 계속 쌓는다고 가정해보자. 여기에 층을 계속 쌓으면 $y(x) = f(f(f(x)))$ 가 될 것인데, 이는 $W \times W \times W \times x$ 이므로 W^3x 일 것이다. 그런데 이는 W^3을 W' 라고 정의하여 $f(x) = W'x$ 라는 하나의 Layer로 표현한 것과 같은 것이다.

즉, 선형 함수로는 은닉층을 여러번 추가하더라도 1회 추가한 것과 차이를 줄 수 없다.

대표적인 활성화 함수로 Sigmoid, tanh, ReLU라는 것이 있고 (딥러닝 클린업에서 자세히 다룰 예정), Sigmoid 함수만 보더라도 $\frac{1}{1+e^{-x}}$ 로, 복잡한 함수의 개형을 가져, 미분하여 0이 되는 값을 찾는 것이 어려워진다. 그렇기 때문에 Gradient Descent와 같은 알고리즘을 사용하여 최적화를 진행하는 것.

---

2. 라그랑주 승수법(Lagrange Multiplier Method)

※ 사실 라그랑주 승수법을 깊게 이해하기 위해서는 Linear Programming과 Duality, 상한과 하한 등에 대한 이해가 선행되어야 수월합니다. 그러나 여기서 모든 개념을 다 다루기에는 분량과 시간 상의 제약이 있어, 머신러닝에 있어 라그랑주 승수법이 쓰이는 직관과 큰 흐름만 이해하는 것을 목표로 합니다.

딥러닝의 예시가 조금 길어졌지만, 돌아와서 최적화를 다시 들여다보자. 지금까지 배운 Gradient Descent 알고리즘은 특히 **제약이 없는 상황**에서 잘 작동하는 알고리즘이었다.

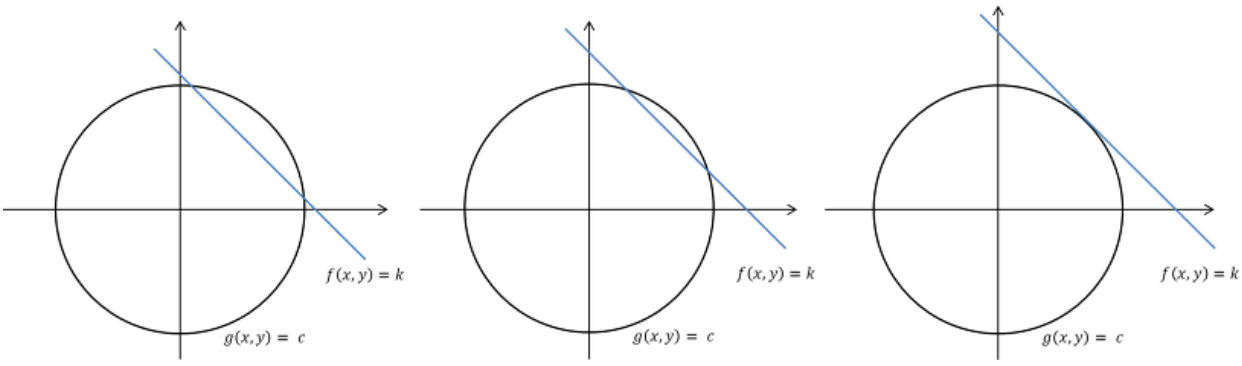
그러나 제약조건이 생긴다면 어떻게 해야 할까? 간단한 예시로 설명해보자면, $y \geq 3$ 이라는 조건 상에서 $(y-1)^2$의 최솟값을 찾고 있는데, 기울기가 0인 지점을 찾으려 했더니 공간이 한정되어 그러한 지점을 찾지 못하는 상황이라고 할 수 있다. 이 때, **라그랑주 승수법**을 이용하여 문제를 풀게 된다.

📌 들어가기 전 핵심 흐름을 정리해보자면

1. 우리가 풀고 싶은 원래 문제**(Primal problem)**가 **제약 조건(Constraints)** 때문에 해결이 어렵다.
2. 원래 문제가 지저분하니, 그 문제와 같은 솔루션을 갖는 다른 문제를 정의하고, 이를 풀어버리자고 생각할 수 있다. 이 다른 문제를 **쌍대 문제(Duality)**라고 한다.
3. 메인문제와 쌍대 문제의 해가 항상 같다는 것은 당연하게도, 보장할 수 없다. 그러나 특정 조건(*Slater's condition)*을 만족할 때 같아질 수 있다.
4. 이 Dual Problem은 당연히 Convex 하길 원하고, 그렇게 만드는 것이 목표이다. 그리고 이 때 사용되는 방법 중 하나가 **라그랑주 승수법**이다.
5. 또한 등식의 제약식이 있는 최적화 문제를 푸는 것을 부등식의 제한 조건에서도 쓸 수 있게 확장시킨 것을 **KKT 조건**이라 한다.

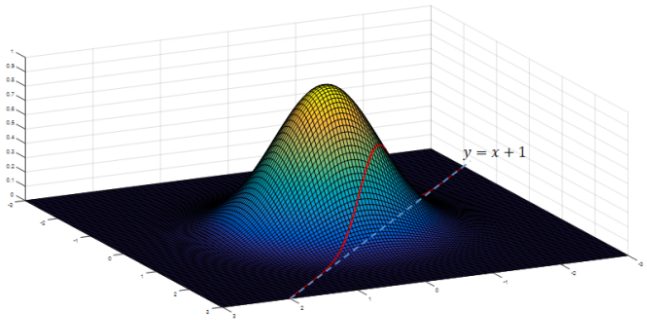
라그랑주 승수법(Lagrange Multiplier Method)

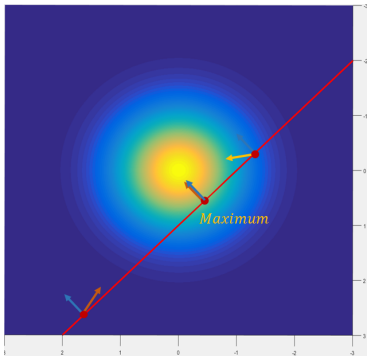
라그랑주 승수법은 등식 제약조건이 있는 최적화 문제를 풀기 위해 고안된 방법으로, 그 풀이는 함수와 제약식의 **공통접선**을 찾는 것이라고도 할 수 있다.



x, y가 원 위에 있다는 제약이 존재하는 상황에서, $f(x, y) = x + y$ 의 최댓값은 어떻게 구할 수 있을까? 오른쪽 그림에서 볼 수 있듯이, 제약식과 $f(x, y)$ 가 접하는 지점이 최댓값일 것이다. 이 접하는 지점을 찾기 위해 기울기가 같다는 것을 이용해, Gradient의 개념을 다시 꺼내오게 된다.

3차원에서도 이해해보자.





다음과 같은 종모양의 곡면 $z = f(x, y)$에서, z의 최댓값을 찾고자 한다. Gradient를 취해 0인 지점을 구하면 되겠지만, $g(x, y) = x - y + 1 = 0$ 이라는 제약이 추가된다면 어떨까?

위에서 쳐다보면서 직선의 법선(파란색), 곡면의 그라디언트 벡터(주황색)를 각각 표시해 본 후 살펴보자. 직선의 gradient(법선)와 곡면의 gradient가 **평행**일 때 최대값이 된다는 것을 직관적으로 볼 수 있다. (직선의 그래디언트는 법선)

위에서 얻어낸 정보를 수식으로 표현하면 다음과 같을 것이다.

$$\nabla f = \lambda \nabla g$$

두 함수를 미분한 기울기에 앞뒤 방향(±)과 길이를 맞추기 위한 미지수 λ를 곱하여, 두 접선이 서로 같다고 놓고 등식을 세운 것이다. 여기서 λ를 라그랑주 승수(Lagrange Multiplier)라고 한다.

Lagrange Duality

라그랑주 쌍대 문제는 최적화하려는 값에 라그랑주 승수(multiplier) 항을 더하여, 제약이 있는 문제를 제약이 없는 문제로 바꾸는 방법이다. 아래의 예시를 살펴보자. 왼쪽의 최적화 문제를 풀기 위해 f, g 의 그래디언트가 평행하다는 식을 세운다.

$$\begin{aligned} &\min\ f(x) \\ &s.t.\ g(x) = 0 \end{aligned} \qquad\qquad \nabla f = \lambda \nabla g$$

그리고 오른쪽의 식을 풀어 쓰면 다음과 같다.

$$\frac{\partial f}{\partial x} - \lambda \cdot \frac{\partial g}{\partial x} = 0, \quad g(x) = 0$$

$g(x) = 0$ 은 제약식을 만족한다는 것을 의미하는 식으로, 그래디언트가 평행인 x를 찾는 것만 달라지고 아직까지 제약식은 남아 있는 형태이다. 아래 두 방정식을 풀면 같은 해가 나온다.

또 이를 두고 식을 바꾸어 문제를 해결할 수도 있는데,

$$\mathcal{L}(x, \lambda) = f(x) + \lambda g(x)$$

라는 라그랑주 함수($\mathcal{L}(x, \lambda)$) 를 정의하는 것이다. 이는 Lower Bound로써 $\min_x \mathcal{L}$ 를 먼저 찾은 후, λ에 대해 다시 그 값을 최대화하는 문제로 바꾸려는 것이다.

이제 이 라그랑주 듀얼 문제는 최소화한 라그랑주 함수를 λ에 대해 최대화하는 것이 목표가 된다. 또한 Slater's condition을 만족했을 때 라그랑주 함수를 λ에 대해 최대화한 값은 원래 문제를 최소화하고자 했던 값과 동일하다.

> 📌 Slater's Condition
>
> 만약 원래(primal) 문제가 convex이고 strictly feasible한 $x \in \mathbb{R}^n$가 하나 이상 있으면 strong duality가 만족.

즉 이로써 등식의 제약이 있는 문제를 **라그랑주 승수법**을 이용해 **제약이 없는 문제**로 바꾸어 문제를 해결할 수 있게 된 것이다.

KKT(Karush-Kuhn-Tucker) 조건

- 등식의 제약식이 있는 최적화 문제를 푸는 라그랑주 승수법을, **부등식**의 제약 조건에서도 쓸 수 있게 확장시킨 것을 **KKT 조건**이라고 한다.
- 이로써 우리는 $\sum \beta_i^2 \leq s$ 와 같이 부등식의 제약이 있는 Ridge Regression 등으로 최적화 문제를 확장할 수 있다.
- 문제를 식으로 나타내면…

$$\min_x f(x) \;\; s.t. \;\; g(x) \leq 0, \; h(x) = 0$$

KKT condition(암기할 필요 X)

1. $\nabla_x f(x) + \mu \nabla g(x) + \lambda \nabla h(x) = 0$ (stationarity)

2. $g(x) \leq 0$
   $h(x) = 0$ (Primal constraints)
3. $\mu \geq 0$ (dual constraints)
4. $\mu g(x) = 0$ (complementary slackness)

이해를 돕기 위해 제약조건을 각각 하나씩만 작성했지만, 다음과 같은 상황도 흔히 볼 수 있다.

$$\nabla_{\mathbf{x}} f(\mathbf{x}) + \sum_{i=1}^{m} \mu_i \nabla_{\mathbf{x}} g_i(\mathbf{x}) + \sum_{j=1}^{k} \lambda_j \nabla_{\mathbf{x}} h_j(\mathbf{x}) = 0$$

하지만 그래봤자 제약들의 선형 결합으로 표현될 뿐, 위에서 언급한 모든 조건들이 단일 조건에서 선형 결합에 대한 조건으로 변경될 뿐이다. 다음부터 진행될 Regularization 방법들을 보면서 '아, **제약을 걸고 최적화를 하는 방법**이고, 결국 **접하는 지점**에서 해를 찾으려는 노력을 하겠구나' 정도로 이해하면 된다.

> ✅ 라그랑주 승수법이 그 목적과 원리에 대해 이해할 필요가 있는 것과 달리 KKT 조건은 필요할 때 찾아보는 정도로, 최적화 문제에서 무슨 조건이 있었지~ 하고 떠올리고 검색할 수 있으면 충분하다.

Ridge Regression

먼저, 라그랑주 승수를 이용한 최적화가 쓰인 가장 대표적 예시로 L2-norm을 이용한 제약을 가하는 **Ridge Regression**을 살펴보고자 한다. 회귀팀 클린업 3주차에서 더 자세히 다룰 예정이기 때문에, Ridge에 대해 생소하다면 '지금까지 배운 게 Ridge라는 것에 쓰이는구나~' 정도로 이해하는 것도 좋다.

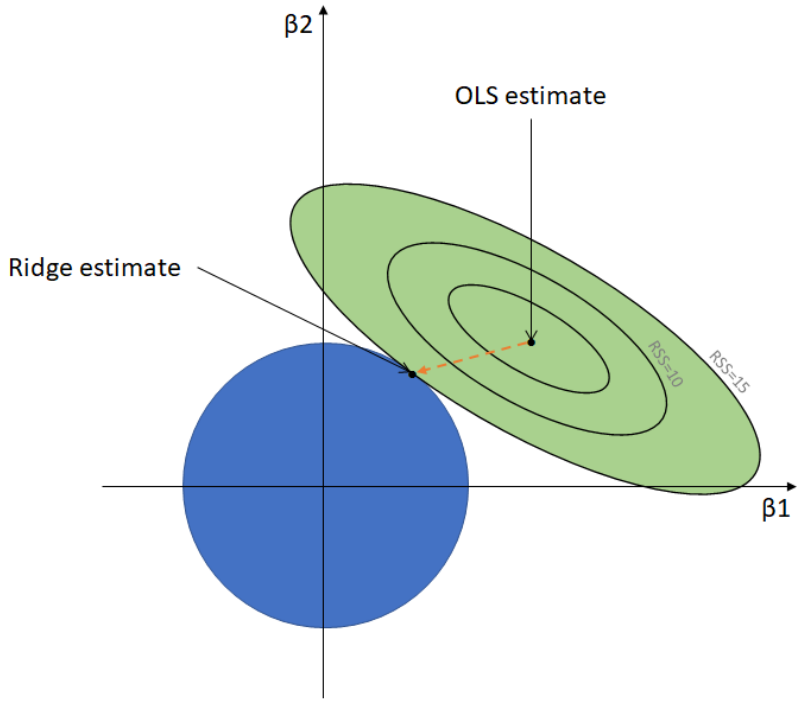
Ridge Regression은 **SSE(Sum of Squares Error)를 최소화하면서 동시에 회귀계수 β의 제곱 합에 대한 제약조건을 거는 방법**이다. 이렇게 하는 이유는 β의 변동성을 줄임으로써, 모델의 분산을 줄이기 위함이다. (데이터가 변해도 추정 식을 크게 변화시키지 않기 위함)

$$\hat{\beta}^{ridge} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - x_i\beta)^2$$
$$\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 \leq s$$

β의 L2 norm을 어떤 범위로 제약을 주는 것인데, 그렇게 되면 이 범위 안에서 최적 β를 구해야 하고, 이는 제약이 있는 최적화 문제이다. 이 문제는 푸는 것이 쉽지 않다. 그러나 이를 라그랑주 승수법으로 표현하면 아래와 같다.

$$\beta^{ridge} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} (y_i - x_i\beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

이렇게 제약이 없는 Ridge regression의 최적화 문제를 완성할 수 있다. 위 식을 최소화하는 β를 구함으로써 Ridge estimator를 구할 수 있고, Ridge 문제의 경우 미분이 가능하므로 미분을 통해서 추정량을 구할 수 있게 된다. 그리고 또한, 최적 파라미터(Ridge Estimator)는 두 식이 접하는 부분(Gradient가 평행해지는 부분)이라는 것도 짚고 넘어간다면 더욱 좋겠다.



이외에도 라그랑주 승수가 머신러닝에 적용되는 사례로는 LASSO, SVM, Spline 등 매우 다양하다. 추후 통계적데이터마이닝 수업을 듣는다면 이와 비슷한 형태가 계속 나타나는 것을 확인할 수 있을 것이다.

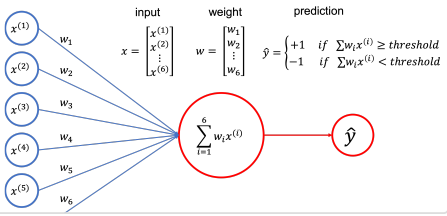
이번 수업은 전공 수업에서 잘 다루지 않았던 딥러닝의 개념과 수학적 원리, 최적화 이론을 다루어 난이도가 높았을 것으로 생각됩니다. 궁금한 점은 팀장이나 학회장팀에게 언제든 질문해주시면 감사하겠습니다. 다음 시간부터는 다시 난이도를 낮추어, 선형대수학의 기본과 통계적 분석에의 적용을 다루겠습니다. 다들 너무 수고하셨습니다~!

Reference

[머신러닝 정리] 퍼셉트론 (Perceptron) - 01. Classification

본 포스팅 시리즈는 다양한 머신러닝 테크닉에 대해 수학적 관점과 실용적 관점에서 정리합니다.

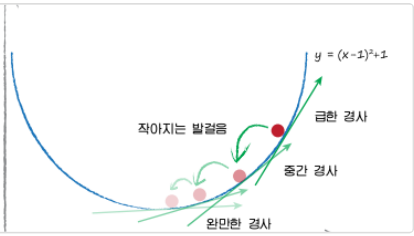
https://velog.io/@shlee0125/머신러닝-정리-퍼셉트론-Perceptron



딥러닝의 핵심, 역전파

역전파(Back Propagation)란 무엇일까? 역전파의 의미 우선 역전파의 정의에 대해서 알아보자. 역전파는 신경망의 각 노드가 가지고 있는 가중치(Weight)와 편향(Bias)을 학습시키기 위한 알고리즘으로, 딥러닝에 있어서 가장 핵심적인 부분이라고

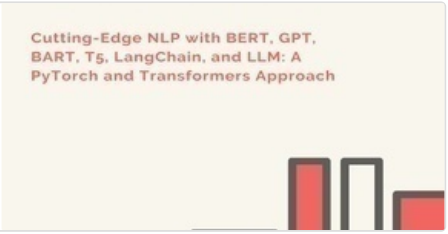
https://re-code-cord.tistory.com/entry/해물파전말고-역전파



PyTorch로 시작하는 딥 러닝 입문

이 책은 딥 러닝 프레임워크 PyTorch를 사용하여 딥 러닝에 입문하는 것을 목표로 합니다. 이 책은 파이썬은 어느정도 할 줄 안다고 가정합니다. 이 이상의 ...

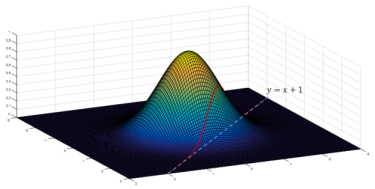
https://wikidocs.net/book/2788



(Lagrange Multiplier)라그랑주 승수법 시각적으로 쉽게 이해하기

다음과 같은 종모양의 곡면이 있습니다. 최대값을 찾으려면 어떻게 해야할까요?? 미적분학 시간에 배웠듯이...

https://m.blog.naver.com/lyb0684/221332307807



모두를 위한 컨벡스 최적화 ·

모두를 위한 컨벡스 최적화

https://convex-optimization-for-all.github.io/

LG Aimers - 주재걸 교수님 딥러닝 관련 강의자료 일부