BOOK : 딥러닝 제대로 시작하기

※ *STEP 1 : START*

- **역전파법**
  - 샘플에 대한 신경망의 오차(error cost, 목표 출력과 실제 출력의 차이)를 다시 출력층에서부터 입력층으로 거꾸로 전파시켜 각 층의 가중치의 기울기를 게산하는 방법입니다.
- **기울기 소실(vanishing gradient)**
  - 역전파법을 수행할 때 입력층에서 멀리 떨어진 **깊은(deep) 층** 에 이르게 되면 기울기가 급속히 작아지거나 너무 커져서 발산해 버리는 현상입니다.
- 90년대 후반 신경망 기법은 보다 나중에 제안된 머신 러닝 기법보다 뒤떨어진다는 평가를 받게 된 이유
  - 기울기 소실 문제가 다층 신경망의 학습을 곤란하게 만들었으며, 신경망은 학습을 위한 여러 파라미터로 층수나 유닛의 수를 갖는데 이 파라미터가 최종적으로 어떻게 성능으로 이어지는지 알 수 없었기 때문입니다.
  - 또한, 층과 층 사이의 결합 밀도가 높은 다층 신경망의 학습이 어려웠다.
- CNN은 특정한 이미지 처리를 수행하는 층이 여러 개 쌓인 구조를 갖는, **층간의 결합 밀도**가 낮은 신경망이다.
- 제프리 힌턴의 연구진이 발표한 **딥 빌리프 네트워크(Deep Belief Network, DBN)** 는 일반적인 신경망과 유사하게 다층 구조를 갖는 그래프 모델(graph model)이다.
  - DBN 모델의 동작은 확률적으로 기술되며, 주로 데이터의 생성 모델로 쓰였다.
  - 다른 신경망과 마찬가지로 층수가 늘어나면 DBN의 학습도 곤란해졌는데 이때 힌턴은 대책으로 **제약 볼츠만 머신(Restricted Boltzmann, RBM)** 이라는 단층 신경망으로 분해한 뒤, 탐욕 알고리즘(Greedy Algorithm)과 같은 아이디어로 이들 RBM에 입력층과의 거리가 가까운 순으로 차례차례 비지도 학습을 수행하는 방법을 제안했다.
- **사전훈련(pre-training)** 이란 사용하려는 신경망을 학습시키기 전에 층 단위의 학습을 거치는 거으로 더 나은 초기값을 얻는 방법입니다.
  - 이후 **DBN**이나 **RBM이** 아닌, 그보다 더 단순한 **자기부호화기(autoencoder)** 를 사용하여도 다층 신경망의 사전훈련이 가능해졌다.
- **오토인코더**
  - 오토인코더란 입력으로부터 계산되는 출력이 입력 자체와 비슷해지도록 훈련되는 신경망입니다.
  - 즉, 오토인코더의 복표 출력은 입력 그 자체이기 때문에 비지도 학습으로 학습이 이루어집니다.
  - **학습 절차** : 목적으로 하는 다층 신경망을 단층으로 분할한 뒤, 입력층과 가까운 순서대로 각 층을 오토인코더로 보고 비지도 학습을 실시해서 각 층의 파라미터의 초기값을 얻습니다. 그 다음, 전체 신경망에 대한 지도 학습을 실시합니다.

※ *STEP 2 : 앞먹임 신경망(feedforward neural network)*

- **앞먹임 신경망**은 층 모양으로 늘어선 유닛이 인접한 층(layer)들과만 결합하는 구조를 가지며, 정보가 입력 측에서 출력 측으로 한 방향으로만 흐르는 신경망입니다. 앞먹임 신경망은 **다층 퍼셉트론(multi-layer)** 이라 부르기도 합니다.

- **바이어스**는 의미 그대로 어떤 '편견'을 학습 결과에 반영시키는 역할을 합니다.

    - 신경망의 유닛 내에서 활성화 함수를 왼쪽 혹은 오른쪽으로 이동시켜 변화시키는 것과 같은 결과를 얻게 됩니다.
    - 이와 달리 **가중치**는 활성화 함수의 곡선을 좌우로 늘리거나 줄이는 것과 같은 결과를 얻습니다.

- 즉, **가중치(weight)** 는 입력신호가 결과 출력에 주는 영향도를 조절하는 **파라미터(매개변수)** 이고, **편향(bias)** 은 뉴런(또는 노드; x를 의미)이 얼마나 쉽게 활성화(1로 출력; activation)되느냐를 조정하는 (adjust) **파라미터(매개변수)** 이다.

- **활성화함수(Activation Function)** 는 말 그대로, 출력값을 활성화 여부를 결정하고, 그 값을 부여하는 함수라 할 수 있다.

    - 활성화 함수를 사용하는 이유는 무엇일까?
    - 정답을 바로 말하자면 활성화 함수 사용의 이유는 **Data를 비선형으로 바꾸기 위해서**이다.
    - 일반적으로 선형 시스템을 사용해야 예측이 가능하고 장점도 많다고 알고 있다. **하지만 선형시스템을 망에 적용시, 망이 깊어지지 않는다.**
    - 선형 시스템의 경우 망이 아무리 깊어지더라도, 1층의 은닉층으로 구현이 가능하다.
    - 망이 깊어지지 않는게 왜 문제가 될까? [망이 깊어질 수록 장점이 많아진다는 것이다.]
        - **망이 깊어진다는 것은 은닉층의 수가 많아진다는 것이다.**
        - **● 망이 깊어질때의 장점**
        - 1. 매개 변수가 줄어든다. : 망이 깊어지면 같은 수준의 정확도의 망을 구현하더라도 **파라미터(매개변수)가 더 적게 필요하다.**
        - 2. 필요한 연산의 수가 줄어든다. : 예를들어 은닉층에 Filter를 Convolution할 때 필터의 크기를 줄이고, 망을 깊게 만들면(은닉층의 수를 많게 만들면) 연산 횟수가 줄어들면서도 정확도를 유지하는 결과를 내는것을 확인할 수 있다.
        - ※이처럼 망이 깊어질수록 효과를 내기 때문에 Deep-Learning이라고도 한다고 들었다.
        - 선형시스템의 경우에는 망이 깊어지지 않으므로, 망이 깊어지는것에 대한 장점을 활용할 수 없다.
        - 이때, 활성화 함수를 사용하면 입력이 들어갈때, 출력값이 선형으로 나오지 않으므로, 망을 깊게 만들 수 있다.
    - 참고 : https://m.blog.naver.com/PostView.nhn?blogId=worb1605&logNo=221187949828&proxyReferer=https%3A%2F%2Fwww.google.com%2F

- 활성화 함수의 종류

    - 1.**시그모이드 함수(Sigmoid Function)=로지스틱 함수** : `h(x) = 1/(1+e^-x)`
        - x값이 작아질수록 0에 수렴하고, 커질수록 1에 수렴한다.
        - 쌍곡선 정접함수 `tanh(x)`도 로지스틱 함수와 비슷한 성질을 갖고 있어 사용가능하나 함수의 치역은 (-1,1)로 달라집니다.
    - 2.**계단 함수(Step Function)** : `y = 1 (when x≥0), 0 (when x<0)`
        - Step Function은 말 그대로 계단모양 함수로, 특정값 이하는 0, 이상은 1로 출력하도록 만들어진 함수이다.
    - 3.**ReLU Function(Rectified Linear Unit)==램프 함수** : `y = x (when x≥0), 0(when x<0)`
        - ReLU 함수는 입력이 특정값을 넘으면 입력이 그대로 출력으로 나오고, 0을 넘지 않을시 0을 반환하는 함수이다.
        - 장점1 : 단순하고 계산량이 적으며, 학습이 빠르다. 최종 결과 또한 더 좋은 경우가 많다.
        - 장점2 : 시그모이드 하수를 사용할 때는 입력값의 변동 폭에 주의할 필요가 있지만(너무 커질 경우 출력 대부분이 0이나 1 둘 중 하나가 됨.), ReLU 함수를 사용하면 그런 일이 일어나

지 않는다.
    - 4.**맥스아웃(maxout) 함수**
        - 이 활성화 함수를 갖는 유닛 하나는 K개의 유닛을 하나로 합친 것과 같은 구조를 갖습니다.
        - K개 하나하나가 서로 다른 가중치와 바이어스를 가지므로 각각의 총 입력을 따로 계산하고 그중 최대값을 이 유닛의 출력으로 합니다.
        - 성능은 좋지만, 파라미터 수가 일반적인 유닛의 K배이기 때문에 그리 많이 쓰진 않습니다.

- 활성화 함수 **책내용**
    - 유닛의 활성화 함수로는 통상적으로 **단조증가(monotone increasing)하는 비선형함수** 가 사용됩니다.
    - 위의 활성화 함수말고도 **선형사상(linear mapping)** 혹은 **항등사상(identity mapping)** 이 있습니다.
    - 신경망에서는 각 유닛의 활성화 함수가 비선형성을 갖는 것이 중요하지만, 부분적으로 선형사상을 사용하는 경우가 있습니다.
        - 예를 들어, **회귀 문제(regression problem)** 를 위한 신경망에서는 출력층에서 **항등사상**을 사용하고, **클래스 분류를 목적**으로 하는 신경망에서는 출력층의 활성화 함수로 대게 **소프트맥스 함수**를 사용합니다.

- **퍼셉트론(Perceptron)이란**
    - 퍼셉트론은 다수의 신호(Input)을 입력받아서 하나의 신호(Output)을 출력하는 인공신공망 모델입니다.
    - 퍼셉트론의 한계는 선형으로 분류를 할 수 있지만 XOR와 같이 **선형 분류만 가능하며 비선형 분류는 불가능하다는 점**이다. 그래서 이후 다층 퍼셉트론이 등장한다.

- 훈련 샘플(training samples)과 훈련 데이터(training data)
    - 입출력 쌍 (x,d) 하나하나를 훈련 샘프이라 부르고, 그 집합을 훈련 데이터라고 합니다.

- **오차함수(error function) == 손실함수(loss function)** 와 학습
    - 모든 입출력 쌍의 입력 x_n에 대한 신경망의 출력(y)이 최대한 d_n과 가까워지도록 w를 조정하는 것이 학습입니다.
    - 또한, 신경망이 나타내는 함수와 훈련 데이터와의 가까운 정도를 나타내는 거리의 척도는 **오차함수**나 **손실함수**라고 부릅니다.
    - 중요! : 문제의 유형에 따라 사용되는 활성화 함수 및 오차함수의 종류
        - 문제 유형 | 출력층에 쓰이는 활성화 함수 | 오차함수
        - 회귀 - **tanh(x)** [치역이 -1 ~ 1인 경우] or **항등사상** (목표함수의 치역이 임의의 실수일 경우) - 제곱오차
        - 이진 분류 - **시그모이드(로지스틱) 함수** - 최대우도법을 사용한 음의 로그우도 : 우도를 가장 크게 하는 w를 선택(+우도의 로그화를 통해 부호를 바꿈)
        - 다클래스 분류 - 소프트맥스 함수 - 교차 엔트로피(=음의 로그우도)

- **교차 엔트로피**란?
    - Cross-entropy는 틀릴 수 있는 정보 양을 고려한 최적으로 인코딩할 수 있게 해주는 정보량으로 생각할 수 있습니다.
    - 사후확률를 활용해 훈련 데이터에 대한 w의 우도를 구하고 로그를 취하여 부호를 반전한 것이 교차 엔트로피입니다.

    - cross entropy는 label의 값이 one_hot encoding (원핫 인코딩) 일 때만 사용이 가능하다는 것을 염두해두자.
    - MSE와의 차이점
        - MSE는 틀린 샘플에 대해 더 집중하는 특성을 가진다. 맞은 것과 틀린 것에 똑같이 집중해야 하는데 그렇지 않아 오차 정의로는 적절하지 않다.
        - 중요한건, weight update를 해야하는데 MSE를 cost function으로 쓰니까 업데이트 되는 양이 너무 적더라..... 해서 생긴 새로운 cost function이 Cross function입니다.
    - 교차 엔트로피의 장점
        - 크로스 엔트로피를 쓰면 딥러닝 역전파시 그래디언트가 죽는 문제를 어느 정도 해결할 수 있고, 그래디언트를 구하는 과정 역시 비교적 간단하다고 합니다.

- **소프트맥스 함수(Softmax)** 란

    - 소프트맥스는 입력받은 값을 출력으로 0~1사이의 값으로 모두 정규화하며 출력 값들의 총합은 항상 1이 되는 특성을 가진 함수이다.
    - 소프트맥스 층은 클래스에 속할 확률을 수치 p로 표현합니다.[총합은 1]
    - **다른 활성화 함수와의 차이점**
        - 그 외의 활성화 함수에서 유닛 k의 출력 z_k는 해당 유닛으로 들어오는 총 입력 u_k로부터만 결정되는 것과 대조적으로, 소프트맥스 함수에서는 이 층 모든 유닛의 총 입력 u1,...,uk으로부터 결정되는 점이 다른 함수들과 다르다.
    - 소프트맥스 함수의 잉여성으로 인해 가중치를 학습할 때 출력층의 가중치가 잘 수렴하지 않고 학습이 느리게 진행된다는 단점이 있지만, 이를 출력층에 가중치 감쇠 등의 제약을 추가해 해결하면 됩니다.

- Q. 기본적인 신경망 계산 과정을 간단하게 설명해주세요

    - 훈련 샘플을 입력하는 feed-forward(앞먹임) 계산을 하고 오차를 구한 뒤, 이어 역전파 계산을 하고 오차의 기울기를 구하여 가중치를 업데이트합니다.

※ *STEP 3*

- Q. **경사 하강법(gradient descent method)** 가 무엇이며 왜 하는 것이죠?

    - 오차함수 E(w)는 일반적인 경우 볼록함수(convex function)가 아니므로 **전역 극소점(global minimun)** 을 직접 구하는 것은 통상적으로 불가능합니다.
    - 그래서 그 대신 오차함수의 **국소 극소점(local minimum)** w를 구하는 것을 생각하게 된 것입니다.
    - 하지만, 이렇게 구한 극소점 중 하나가 우연히 전역 극소점일 가능성은 높지 않지만 오차함수가 충분히 작다면 괜찮습니다.
    - local minimum 하나는 어떤 초기점을 출발점으로 하고 w를 되풀이하여 갱신하는 반복 계산을 통해 구할 수 있습니다.
    - 그 중에서 가장 간단한 방법이 **경사 하강법(gradient descent method)** 입니다.
        - 경사 하강법은 현재의 w를 음의 기울기 방향으로 조금씩 움직이는 것을 여러 번 반복하는 방법입니다.

- Q. **확률적 경사 하강법(Stochastic gradient descent)** 란?

    - 각 샘플 한개에 대해서만 계산되는 오차의 합 **(배치학습)** 이 아니라 샘플의 일부를 사용하여 파라미터를 업데이트 하는 방법입니다.

    - Q-1. 신경망 학습에서 배치 형태의 경사 하강법보다 확률적 경사 하강법이 더 많이 쓰이는 이유는?
        - 훈련 데이터에 잉여성이 있을 때 계산 효율이 향상되고 학습이 빠릅니다.
        - 반복 계산이 바람직하지 않은(오차함수의 값이 상대적으로 작은) 국소 극소점(local minimum)에 갇히는 위험을 줄일 수 있습니다.
        - 배치 학습의 경우 최소화하려는 목적함수는 항상 같은 오차함수 E(w)이므로, local minimum에 한번 갇히면 두 번 다시 빠져나올 수 없게 됩니다.
        - 하지만, 확률적 경사 하강법은 w를 업데이트할 때마다 목적함수 E(w)가 달라지므로 위의 위험이 적어지게 됩니다.
        - 또한, 파라미터의 업데이트 크기가 작은 상태로 학습이 진행되므로 학습의 경과를 좀 더 자세히 들여다보면서 진행할 수 있다는 점과 **온라인 학습**, 즉 훈련 데이터의 수집과 최적화가 동시에 진행될 수 있다는 점이 있습니다.

- Q. **과적합(overfitting)** 이란 무엇인가?

    - A. 과적합이란, 학습 시에 오차함수의 값이 작은 극소점(local minimum)에 갇힌 상황이다.
    - B. 훈련 오차와 테스트 오차가 달라지게 되는 경우를 뜻하기도 합니다.

- Q. 과적합을 **완화**시키는 방법에는 무엇이 있죠?

    - A. 과적합을 완화시키려면 **규제화(regularization)** 를 해야하는데 방법으론 `가중치 감쇠(weight decay), 가중치 상한, 드롭아웃(drop-out)` 등이 있어요.
    - Q-1. 왜 위와 같은 방법들이 제안되었어요?
        - 신경망의 자유도를 낮추는 것은, 과다한 자유도를 가질 경우를 제외하고는 그다지 바람직하지 않아요. 그래서 학습 시에 가중치의 자유도를 제약하는 규제화에 의해 과적합 문제를 완화하도록 몇 가지 방법이 제안됐어요.
    - Q-2. **가중치 감쇠(weight decay)** 는 무엇이죠?
        - 가장 간단한 규제화 방법은 가중치에 어떤 제약을 가하는 거에요. 그중에서도 오차함수에 **가중치의 제곱합(norm의 제곱)** 을 더한 뒤, 이를 최소화하는 방법이 가중치 감쇠에요.
        - 추가로, **람다(λ)** 는 이 규제화의 강도를 제어하는 파라미터인데요, 이 람다를 추가하여 가중치는 자신의 크기에 비례하는 속도로 항상 감쇠하도록 업데이트돼요.
    - Q-3. **가중치 상한**은요?
        - 각 유닛의 입력 측 가중치에 대해서 그 제곱합의 최댓값을 제약하는 방법이에요.
        - 만약에 가중치 상한 부등식을 만족하지 않는다면, 가중치에 미리 정한 (1보다 작은) 상수를 곱하여 부등식을 만족하도록 만들어요.
    - Q-4. 마지막으로 **드롭아웃(drop-out)** 이 무엇이죠?
        - 다층(multi-layer) 신경망의 유닛 중 일부를 확률적으로 선택하여 학습하는 방법이에요.
        - 자세히 말하면, 중간층과 입력층 각 층의 유닛 중 미리 정해 둔 비율 p만큼을 선택하고 선택되지 않은 유닛을 무효화 취급합니다.
        - 즉, 가중치를 업데이트할 때마다 다시 무작위로 선택되며 학습이 끝나고 추론 시에는 모든 유닛을 사용하여 feed-forward(앞먹임) 계산을 합니다.
        - 드롭아웃의 대상이 되었던 층의 유닛은 모든 출력을 p배로 합니다.(또는 출력의 가중치를 p배로 해요.)
            - 그 이유는 추론시의 유닛 수가 학습 시에 비해 1/p배 된 것과 같기 때문에 이를 보상하기 위함이에요.
        - 특히, 클래스 분류의 출력층에 사용되는 소프트맥스 층에 대해서는 드롭아웃 방법이 출력의 기하평균을 내는 것과 같음을 알 수 있어요.
    - Q-5. RBM이나 오토인코더의 학습에 드롭아웃을 적용하면 어떤 효과를 얻을 수 있죠?

        - 두 경우 모두 희소적 특징이 자동 학습된다는 결과가 보고되었어요. 즉, 희소 규제화를 거치지 않고도 유사한 특징을 얻을 수 있다는 이야기죠.
    - Q-6. 드롭아웃과 비슷한 방법이 무엇이 있죠?
        - 드롭 커넥트와 확률적 최대 풀링이 있어요.

- Q. **데이터 정규화(normalization of data)** 가 뭐에요?

    - 각 샘플 Xn을 **선형 변환**하는건데 (Xn-X평균)/표준편차로 구할 수 있어요.
    - 데이터 정규화를 거친 샘플은 각 성분의 평균이 0, 분산이 1이 돼요.
    - 특정한 성분의 분산이 0이거나 매우 작은 경우가 있으면 max(표준편차,충분히 작은값)으로 나누어줘요.

- Q. **데이터의 확장(data augmentation)** 이란 무엇이죠?

    - 데이터 확장은요 샘플 데이터를 일정하게 가공해서 양적으로 '물타기'하는 작업이라고 해요.
    - Q-1. 이걸 왜 하는거죠?
        - 확장을 하는 이유는 트레이닝 데이터의 부족이 **과적합**을 일으키는 가장 큰 원인이기 때문이에요.
    - 이미지 데이터같이 샘플의 분포 양상을 예상할 수 있는 경우에 특히 유효해요. 예를 들어서 같은 카테고리의 물체 이미지이기만 하면 어느 정도 변화를 가해도 타당한 훈련 샘플이라고 보기 때문이에요.
    - 가우스 분포를 따르는 **랜덤 노이즈**를 일괄적으로 적용하는 방법도 유효한 방법이에요.

- Q. **학습률(Learning Rate)** 이 무엇이며 그것이 왜 중요한가요?

    - 경사 하강법에서는 **파라미터의 업데이트 정도**를 학습률을 통해 조절해요. 학습률 설정은 학습의 성패를 좌우할 정도로 중요해요.
    - Q-1. 학습률은 어떻게 결정하나요?
        - 학습률을 결정하는데에 정석과 같은 두 가지 방법이 있어요.
        - 첫 번째는, 학습 초기에 값을 크게 설정했다가 학습을 하면서 학습률을 점점 줄여가는 방법이에요.
        - 두 번째는, 신경망의 모든 층에서 같은 학습률을 사용하는 것이 아니라 층마다 서로 다른 값을 사용하는 방법이에요.
            - 예를 들어, 출력 방향에 가까운 얕은 층에서는 학습률을 작게 잡고, 입력에 가까운 깊은 층에는 크게 잡는 경우가 있어요.
            - 또한, CNN처럼 **가중치 공유**를 할 때 학습률을 가중치가 공유되는 수의 제곱근에 비례하도록 설정하면, 가중치의 업데이트 속도를 대체로 비슷하게 맞출 수 있어요.

- Q. 위의 질문과는 달리 학습률을 자동으로 결정하는 방법에는 뭐가 있죠?

    - AdaGrad가 있어요. 이거는 직관적으로, **자주 나타나는** 기울기의 성분보다 **드물게 나타나는** 기울기 성분을 더 중시해서 파라미터를 업데이트하는 방법이에요.

- Q. **모멘텀(momentum)** 을 설명해주세요.

    - 모멘텀은 경사 하강법의 **수렴 성능을 향상시키기 위한 방법** 중 하나인데요, 가중치의 업데이트 값에 이전 업데이트 값의 일정 비율을 더하는 방법이에요.
    - Q-1. 모멘텀의 효과는 무엇이죠?
        - 오차함수가 깊은 골짜기 같은 형상을 가지는데, 골짜기 바닥이 평평할 때 약간만 빠져나와도 골짜기와 직교하는 방향으로 큰 기울기가 생기면서 바닥을 정상적으로 탐색하지 못하는

문제가 있어요
        - 이러한 문제를 해결하고 골짜기 방향을 따라 골짜기 바닥을 효율적으로 탐색할 수 있게 만들어주죠.

- Q. **가중치의 초기화** 하는 방법을 말해주세요.
    - 가장 일반적인것은 가우스 분포로부터 랜덤값을 생성하여 초기값으로 삼는 방법이에요. 그리고 가우스 분포의 표준편차 선택도 중요한데, 보통 표준편차를 크게 잡으면 초기 학습은 빠르게 진행되지만, 오차함수의 감소가 일찌감치 멈춰버리는 경향이 있을 수 있어요.

*※ STEP 4 : 역전파 * [책보기]

- Q. **역전파법(backpropagation)** 이 뭔가요?
    - 역전파는 앞먹임 신경망 학습에서 가중치와 바이어스에 대한 오차함수의 미분을 계산해야하는데 이러한 미분을 효율적으로 계산하는 방법이에요.
    - Q-1. 역전파법을 왜 사용하죠?
        - **경사 하강법**을 실행하기 위해서는 오차함수 E(w)의 기울기를 계산해야 하는데, 이 미분의 계산이 매우 까다롭기 때문에 역전파법을 사용하는거죠.
        - 각 층의 결합 가중치(w)와 각 유닛의 바이어스(b)에 대한 오차함수의 편미분이 기울기 벡터의 각 성분이고, 자세히는 중간층, 특히 입력이 가까운 깊은 층의 파라미터일수록 미분을 계산하기 까다로워요.
- Q. 오차 역전파를 통해 **오차 기울기(가중치에 대한 오차의 미분)를 계산하는 절차** 를 말해줘요.
    - 1. 각각의 층의 유닛 입력 u과 출력 z을 순서대로 계산한다.
    - 2. 출력층 델타(δ)를 구한다. (통상적으로 δ = z - d : 출력층 L의 유닛 j의 델타 δ는 신경망의 출력(z)과 목표 출력(d)의 차가 된다.)
    - 3. **역전파** : 각 중간층 l(= L-1, L-2, .., 2)에서의 델타 δ를 출력층부터 가까운 순서대로 계산한다.
    - 4. 각 층 l(= 2, ..., L))의 파라미터 w에 대해 미분을 계산한다.
    - **참조** : l-1번째 층의 유닛 i와 l번째 층의 유닛 j를 잇는 결합의 가중치 w_ji에 대한 미분은, 유닛 j에 대한 **델타(δ_j)(L)** 와 유닛 i의 **출력 z_i(L-1)** 의 **곱**에 지나지 않는다.
- Q. 순전파와 역전파 계산의 **공통점과 차이점**은?
    - **공통점** : 순전파와 역전파 계산은 모두 층 단위의 행렬 계산으로 나타낼 수 있으며 식의 형태가 닮았다는 공통점이 있다.
    - **차이점** : 순전파는 **비선형 계산**인데 비해, 역전파는 **선형 계산**이라는 차이점이 있다.
        - 순전파 계산에서는 각 층에 대한 입력은 유닛이 갖는 활성화 함수를 경유하기 때문에, 활성화 함수가 비선형이라면 이 층의 입출력의 관계도 비선형성을 갖는다.
            - ex) 로지스틱 함수를 예로 들면 각 층의 출력은 항상 [0, 1]의 범위로 제약되며, 값이 지나치게 커져서 발산해 버리는 일은 일어나지 않는다.
        - 한편, 역전파 계산은 선형 계산이다. 그 결과, 각 층의 가중치의 값이 크면 델타가 각 층을 거쳐 전달되는 도중에 **급속하게 커지거나(발산)**, 혹은 반대로 기울기가 작으면 **급속하게 작아져 0(소실)** 이 되어 버린다. 어떤 경우든 가중치의 업데잍트가 잘안되며 학습 자체가 어려워진다.

※ STEP 5 : 자기부호화기(autoencoder)

- Q. 오토인코더의 목적이 뭐에요?
    - 오토인코더는 **목표 출력없이 입력만으로** 구성된 트레이닝 데이터로 **비지도 학습**을 수행하여 데이터의 특징을 나타내는, 더 나은 표현을 얻는 것이 목표인 신경망입니다.
    - 또한, 딥 네트워크의 사전훈련(pre-training), 즉 그 **가중치의 좋은 초기값**을 얻는 목적으로 이용됩니다.
    - 자세히 설명하면, feature의 학습을 통해 샘플 x의 또 '다른' 표현인 y를 얻는 것이고, 직관적으로 x를 그대로 쓰는 대신 변환된 y를 사용하는 것입니다.

- Q. 오토인코더의 학습 목표가 뭐죠?
    - 오토인코더의 학습의 목표는 입력을 **부호화(encode)** 한 뒤, 이어 다시 **복호화(decode)** 했을 때 원래의 입력을 되도록 충실히 재현할 수 있는 부호화 방법을 찾는 것이에요.
    - 또한, 오토인코더는 **오차함수를 최소화하는 과정**을 통해 신경망의 가중치와 바이어스를 결정하게 됩니다.

- Q. 오토인코더의 **활성화 함수**와 **오차함수**는 보통 무엇을 사용하죠?
    - 오토인코더의 활성화 함수중에서 중간층의 f는 자유롭게 바꿀수 있으며 통상적으로 **비선형함수**를 사용합니다. 그리고 출력층의 f'은 신경망의 목표 출력이 입력한 x 자신이 될 수 있도록 입력 데이터의 유형에 따라 선택합니다.
    - 1.x의 성분이 실수값을 가질때, 출력층의 f'은 항등사상으로 하는 것이 좋고 오차함수로는 입력값과 출력값에 차에 대한 제곱합을 사용합니다.
    - 2.x의 성분이 이진값을 갖는 경우에는 f'으로 로지스틱 함수를 보통 사용하고 오차함수로는 교차엔트로피를 사용합니다.

- Q. 오토인코더를 결정하는 요소에는 뭐가 있을까요?
    - 주로 중간층의 유닛 수와 해당층에서 사용되는 활성화 함수가 있죠.

- Q. 오토인코더의 최적화에 대해서 주의할 점을 간단히 설명해주세요.
    - 오토인코더의 최적화는 확률적 경사 하강법의 샘플 추출 시를 제외하면, 랜덤성 없이 결정론적으로 진행된다는 점에 주의하면 됩니다.

- Q. **과완비(overcomplete)** 란 무엇이죠?
    - 오토인코더는 입력 데이터의 **feature(자질)** 를 학습함으로써, 입력 데이터에서 불필요한 정보를 제거하고 그 본질만을 추출하는 겁니다.
    - 이러한 이유 때문에 입력 데이터의 성분 수 Dx보다도 인코딩된 부호가 갖는 성분 수 Dy는 자연히 더 작을 거라고 생각하는데 항상 그렇지만은 않아요.
    - 즉, 희소 규제화를 이용한다면 여분의 자유도를 갖는 특징이어도 입력 데이터를 잘 나타내는 자질을 얻는 것이 가능해지는데, 이것을 **과완비(overcomplete)** 한 표현이라고 합니다.

- Q. 그럼 **희소 오토인코더(sparse autoencoder)** 는요?
    - 예를 들어서 중간층에 활성화 함수로 선형함수를 사용한 경우, 중간층의 유닛수 Dy가 입력층의 유닛 수 Dx보다 많다면 무의미한 결과밖에 얻을 수 없습니다. 활성화 함수에 비선형함수를 사용하면 이러한 논의는 처음부터 성립할 수 없지만, 중간층의 자유도가 입력의 자유도보다 크다는 것은 변하지 않으며, 쓸모없는 해만을 얻게 될 가능성이 높습니다. 이에 대해 희소 규제화를 사용하면, 증

간층의 유닛 수가 더 많은 경우(Dy >= Dx)에도 오토인코더가 의미 있는 표현을 학습할 수 있게 되는데 이를 **희소 오토인코더**라고 부릅니다.

- Q. **희소규제화**에 대해서 좀 더 자세히 설명해주세요.
    - 기본적인 아이디어는 훈련 샘플 Xn을 되도록 적은 수의 중간층 유닛을 사용하여 재현할 수 있도록 파라미터를 결정하는 것입니다.
    - 또한, 입력 x로부터 중간층의 출력 y를 거쳐 출력 x'가 계산되는 과정에 있어서, y의 각 유닛 중 되도록 적은 수의 유닛만이 0이 아닌 출력치를 갖고 나머지는 출력이 0이 되도록(=발화하지 않음)하는 제약을 가합니다.
        - 원래의 오차함수 E(w)에 희소 규제화 항을 추가한 E'(w)를 최소화한다.
    - 간단히는 활성도의 평균값이 작아지도록 제약을 가하고 각 샘픔을 표현하는 데 쓰이는 중간층 유닛의 수가 적어지도록 학습을 진행한다고 설명할 수 있습니다.

- Q. 희소규제화에서 **로(ρ)** 와 **베타(β)** 는 무엇을 의미할까요?
    - 우선 로(ρ) 햇은 중간층의 유닛의 평균 활성도의 추정치를 나타내고 로(ρ)는 그 목표치가 되는 파라미터이다.
    - 원래의 오차함수 E(w)에 최소 규제화 항을 추가한 E'(W)를 최소화하면 중간층의 각 유닛 평균 활성도가 작은 ρ에 가깝게 되고, 입력의 재현오차 E(w)가 작아지도록 w가 정해진다.
    - 베타(β)는 이 두가지 목표의 밸런스를 결정하는 파라미터다.

- Q. 그렇다면 희소규제화 항에서의 **베타(β)** 에 따라 어떤 특징이 나타날까요?
    - MNIST를 예로 들면 β가 0, 즉 희소 규제화를 하지 않은 경우에 학습된 특징은 어수선한 패턴을 보입니다.
    - 이에 반해 β = 3.0 정도의 강한 규제화를 한다면 각각의 숫자가 그대로 특징으로 선택되고 맙니다.
    - 즉, 학습 시의 희소 규제화는 오토인코더의 중간층의 각 유닛을 '분담'하여 표현하는 양상을 제어하는 역할을 한다고 할 수 있습니다.
    - 희소 규제화를 하지 않을 때는 중간층의 유닛은 각각 제멋대로 입력을 표현하려고 하지만, 알맞은 정도로 희소 규제화가 행해지면, 입력이 갖는 구조를 효율적으로 표현할 수 있도록 중간층의 유닛이 협력하여 각각의 입력을 표현하게 됩니다. 또, 희소 규제화가 너무 강하면 중간층 유닛이 집합으로서가 아닌, 되도록 단독으로 각각의 입력을 표현하려고 하는 경향이 있습니다.

- Q. **희소규제화**와 **가중치 감쇠**의 차이점은?
    - 가중치 감쇠의 규제화 항은 가중치 파라미터 그 자체에 대한 함수이므로 가중치의 업데이트 식만 수정하면 된다.
    - 하지만, 희소 규제화 항의 경우는 **해당 층 유닛의 활성도에 대한 제약**이기 때문에, 단지 그 층뿐 아니라 해당 층 아래에 있는 모든 층의 파라미터에 의해 정해지게 된다.
    - 따라서 일반적으로는 델타의 역전파 자체를 수정해야 한다는 차이점이 있다.

- Q. 배치 최적화와 미니배치에서 평균 활성도를 어떻게 구할까요?
    - 배치 최적화에서는 모든 샘플의 feed-forward 계산을 한 번 해서 각 유닛의 출력을 구한 뒤 활성도의 평균을 계산해야 합니다.
    - 하지만 미니배치를 사용하여 학습 중이라면, 평균 활성도만을 계산하기 위해 모든 샘플을 계산하는 것이 비효율적이므로 미니배치 내의 모든 샘플에 대해서만 평균 활성도를 구하는 것을 반복합니다.

- Q. **데이터의 백색화(whitneing)** 가 무엇입니까?
    - Trainning 데이터의 불필요한 경향은 학습을 방해하는 원인이 되는데, 학습 전 이를 처리하는 것이 **백색화(whitening)** 라고 합니다. (정규화와 비슷하지만 수준이 더 높음)
    - 백색화는 오토인코더가 좋은 자질을 학습할 수 있을지를 크게 좌우할 수 있습니다.
    - Q-1. 그럼 백색화를 왜 할까요?
        - 훈련 샘플에서 성분 간의 상관성을 제거하려고 백색화를 합니다.
    - Q-2. 정규화랑은 무슨 차이일까요?
        - 정규화는 단위 처리였던 데에 비해, 백색화는 성분 간의 관계를 수정하는 처리입니다.

- Q. **PCA 백색화**와 **ZCA 백색화**는 무엇인가요?
    - 공분산행렬의 고유벡터를 이용하는 것이 샘플 집합에 대한 주성분 분석(PCA)과 일맥상통한다는 점에서 P를 PCA 백색화라고 해요.
    - (P^T)P= Φ를 만족하는 P를 대칭행렬(P=P^T)로 제한하는 방법이 ZCA 백색화라 합니다.
        - 그리고 어떤 백색화를 사용하는 경우에도 데이터에 따라 특정 성분의 분산이 매우 작거나 극단적으로 0이 되는 경우가 있어서 작은 값(ε)을 사용합니다.
        - 조금 더 설명하자면, ZCA 백색화는 각 열벡터(이미지 필터)가 하나하나 다른 픽셀을 샘플로 하고, 그 **픽셀과 주위 픽셀과의 차이를 강조하는 효과(온센터 : on-center)** 를 갖고 있습니다.
        - 데이터 정규화만 했을 때보다 추가적으로 ZCA 백색화를 거치면 훨씬 섬세한 패턴이 학습됩니다.
    - Q-1. PCA 백색화와 ZCA 백색화의 차이점은?
        - ZCA 백색화를 거친 샘플은 직류성분이 제거되어 이미지의 모서리가 강조되어 있는데 반해, PCA 백색화를 거친 샘플은 고주파성분이 이미지 전체로 강조되어 원래 이미지의 공간구조가 거의 남지 않은 것처럼 보여져요.

- Q. 한번 더 물을게요. DNN에서 **사전훈련(pre-training)** 을 왜 하죠?
    - 여러 층으로 구성된 feed-forward 신경망은 기울기 소실(gradient vanishing) 문제로 인해 일반적으로는 학습이 잘 되지 않기 때문에 여러 방법 중 하나로 사전훈련을 합니다.
    - Q-1. **사전훈련(pre-training)** 의 기본적인 아이디어는 뭐죠?
        - feed-forward 신경망의 지도 학습법에서는 일반적으로 학습을 시작할 때의 초기 가중치를 랜덤값으로 초기화하는데, 이 초기값을 좀 더 좋은 값으로 정하는 것을 목표로 해요.

- 다양한 사전훈련(pre-training) 중에서 가장 기본적인 **오토인코더**를 사용한 방법의 **절차**를 설명해주세요.

  ◎ pre-training 과정

    1. 다층 신경망을 한 층씩 여러 개의 단층 신경망으로 분할합니다.
    2. 분할한 단층 신경망을 입력층에서 가까운 순서대로 오토인코더와 같은 방법으로 비지도 학습을 수행하여 각 층의 파라미터를 결정합니다.
        - 구체적으로는, 훈련 데이터 {Xn}을 학습시켜 신경망의 가중치 W^(2)와 바이어스 b^(2)를 결정합니다.
    3. 학습한 파라미터를 이 단층 신경망에 설정한 상태에서 훈련 데이터 {Xn}을 입력하여 그 출력층의 표현 {Zn^(2)}를 구합니다.
    4. 그 인접층을 포함하는 단층 신경망을 마찬가지로 오토인코더를 만들고, 이번에는 {Zn^(2)}를 훈련 샘플로하여 비지도 학습을 수행해 W^(3)와 b^(3)를 학습합니다.

    - 5. 학습한 파라미터를 이 단층 신경망에 설정한 후, {Zn^(2)}를 입력하여 그 출력층의 표현 {Zn^(3)}을 구합니다.
    - 6. 이 과정을 층수만큼 상위층으로 올라가는 순서대로 반복한다. [이 과정을 통해 각 층의 가중치와 바이어스를 얻을 수 있음]
    - => 이러한 오토인코더를 층층이 쌓은 것을 **적층 자기부호화기(stacked autoencoder)** 라고 부릅니다.

  ◎ pre-training 이후 지도 학습

    - 사전훈련을 통해 얻은 파라미터를 원래의 신경망 초기값으로 설정한 뒤, 출력층을 포함하는 새로운 한 개 이상의 층을 신경망의 최상위에 추가한다.
    - 이후, 추가된 층의 가중치는 랜덤하게 초기화한 후에 당초의 목표였던 **지도 학습**을 수행한다.(ex 분류문제라면 출력층의 활성화 함수는 소프트맥스)
    - => 이렇게 사전훈련으로 얻은 파라미터를 초기값으로 사용한다면, 신경망의 가중치를 랜덤값으로 초기화했을 때보다 **기울기 소실 문제**가 발생할 가능성이 훨씬 작아지고 학습이 잘 진행된다.

- 사전훈련이 잘 기능하는 이유는 무엇이라고 생각하시나요?

    - x의 벡터공간 내의 **데이터 {Xn}의 분포**를 오토인코더의 **비지도 학습**으로 잘 포착했기 때문이라고 추측해 볼 수 있습니다

- 그외에 **심층 오토인코더(deep autoencoder** 가 있는데 이게 뭐죠?

    - 단층 feed-forward 신경망이 아니라 더 많은 층으로 구성된 오토인코더를 뜻합니다.

- **디노이징 오토인코더(denoising autoencoder)** 란 무엇인가요?

    - 디노이징 오토인코더는 오토인코더를 확장한 것으로, 학습 시에 확률적인 요소를 도입하여 결과적으로는 RBM에 맞먹는 성능을 가지게 됩니다.
    - 또한, 신경망의 구조 자체는 보통 오토인코더와 완전히 같지만, 학습 방법이 다음과 같이 일부 다릅니다.
        - 첫째, 훈련 샘플 x를 확률적으로(랜덤하게) 변동시켜 x'를 얻습니다.
            - ex) 가우스 분포를 따르는 랜덤 노이즈를 더하여 x' = x + δx 로 만듦
        - 둘째, 이렇게 만든 x'를 오토인코더에 입력하고, 부호화(인코더) 및 복호화(디코더)를 거쳐 얻은 출력이 노이즈를 더하기 전 원래 샘플 x에 가까워지도록 학습을 진행합니다.
        - => **보통의 오토인코더에서는 출력이 입력 자체에 가깝게 되도록 학습하기 때문에 x'가 입력이라면 목표 출력도 항상 x'이어야 합니다.(이 부분이 일반 오토인코더와의 차이점)**
            - **그 이외에는 동일** : 출력층의 활성화 함수가 항등사상 -> 제곱오차를 오차함수로 사용, 시그모이드 함수 -> 교차엔트로피를 오차함수로 사용.
            - 이 오차가 작아지도록 신경망의 가중치와 바이어스를 확률적 경사 하강법으로 구합니다.
            - 이때 항상 xn을 조금 변동시킨 x'n을 제시하면서 원래의 xn이 출력되도록 신경망을 훈련시킵니다.
            - 훈련 후에는 입력을 재현할 수 있는 것뿐만 아니라 **노이즈를 제거하는 능력**을 기대할 수 있으며, 이것이 이 이름의 유래가 되었습니다.

※ *STEP 6 : 합성곱 신경망(Convolutional Neural Network)*

합성곱 신경망의 개요

- 합성곱 신경망은 **합성곱층**과 **풀링층**이라는 특별한 두 종류의 층을 포함하는 feed-forward 신경망으로, 주로 이미지 인식에서 사용됩니다.
- 공통점 : 이전 feed-forward 신경망과 마찬가지로, 역전파법과 확률적 경사 하강법을 사용하여 최적화를 수행합니다.
- 차이점 : 합성곱 신경망의 특징은 **국소 감수 영역(local receptive field)** 및 **가중치 공유**라 불리는 특별한 **층간 결합**을 갖는 것입니다.
    - 이전 feed-forward 신경망은 인접층의 유닛이 모두 서로 연결된(전결합, fully-connected) 것이었다.
    - 이에 비해 CNN은 인접한 층과 층 사이에 특정한 유닛만이 결합을 갖는 특별한 층을 갖게 되고 이러한 층에서 합성곱과 풀링이라는 이미지 처리의 기본적 연산을 하게 됩니다.
- Q. CNN의 전형적인 구조는 어떻게 되죠?
    - Input - **[합성곱층(n개) - 풀링] 1쌍** - LCN(국소 콘트라스트 정규화) - **[합성곱층(n개) - 풀링] 1쌍** - 전결합층(fully-connected layer) - 전결합층 - softmax - Output 입니다.
        - 합성곱층과 풀링층이 나오는 형태로 쌍을 이루며 여러 번 반복됩니다. 그리고 국소 콘트라스트 정규화(LCN)층을 배치하는 경우도 있습니다.
        - 합성곱층과 풀링층이 반복되는 구조 뒤에는 인접한 층 사이의 모든 유닛이 결합한 층이 배치되는데 이를 **전결합층(fully-connected layer)** 라고 하며, 일반 feed-forward 신경망의 각 층과 같습니다.
        - 전결합층은 일반적으로 여러 층을 연결하여 배치하고 마지막 출력층은 일반 feed-forward 신경망과 같습니다. [분류 목적이면 마지막 출력층 softmax]
- Q. **합성곱 계산**은 어떻게 하죠?
    - 이미지의 합성곱이란, 이미지와 필터 사이에 정의되는 연산입니다.
    - 이미지의 픽셀값과 필터(작은 크기의 이미지)의 픽셀값을 곱한 것이 이미지의 합성곱입니다.
    - 즉, 이미지에 필터를 겹쳤을 때, 이미지와 필터가 겹쳐지는 픽셀끼리 곱을 구한 다음 필터 전체에서 그 값을 합하는 연산을 뜻합니다.
    - **참고** : 결과 크기 = 입력크기 - 필터크기 + 1
- Q. **합성곱**은 어떠한 작용을 하나요?
    - 이미지의 합성곱은 필터의 명암 패턴과 유사한 명암 패턴이 입력된 이미지에서 어디에 있는지를 검출하는 작용을 합니다.
    - 간단히 말하면, 합성곱 연산은 이미지 처리에서 말하는 필터연산에 해당합니다.
- Q. **제로 패딩(zero-padding)**은 뭔가요?
    - 제로 패딩을 설명하려면 패딩을 먼저 설명할게요.
    - 보통 합성곱한 결과 이미지가 입력 이미지와 크기가 같으면 편리할 때가 많아요.
    - 입력 이미지의 바깥쪽에 폭만큼 테두리를 둘러 크기를 늘려서 출력 이미지의 크기가 원래 입력 이미지와 같은 크기가 되도록 만드는 것입니다.
    - 그 중에서도 제로 패딩은 이 '테두리' 부분의 픽셀값을 0으로 설정하는 방법입니다.
    - 제로 패딩 처리를 하면 출력 이미지의 주변부가 어둡게 될 수 있어요.

- Q. **스트라이드(stride)**란 무엇인가요?
    - 필터를 적용하는 위치의 간격을 스트라이드(stride)라고 합니다.
    - 스트라이드를 키우면 출력 크기는 작아지지만, 너무 큰 값을 지정하는 것은 이미지의 특징을 놓칠 가능성이 있기 때문에 주의해야 합니다.

- Q. **합성곱층의 특징**이 뭐죠?
    - 국소적인 결합을 가지는 것과 가중치를 공유하는 것입니다.
        - 국소적 결합 : 합성곱 연산의 국소성을 반영하여 출력층의 유닛 하나(채널 m의 한 개 픽셀)는 입력층의 유닛하고만 결합한다는 뜻입니다.
        - 가중치 공유 : 결합의 가중치는 출력층의 같은 채널에 속하는 모든 유닛에서 같다는 뜻입니다.

- Q. **풀링층** 이란 무엇인가요?
    - 풀링층은 보통 합성곱층의 바로 뒤에 배치됩니다.
    - 합성곱층에서 추출한 feature의 위치 감도를 약간 저하시켜 대상이 되는 feature값의 이미지 내에서의 위치가 조금씩 변화하는 경우에도 풀링층의 출력이 변화하지 않도록 해주는 역할을 합니다.

- Q. **최대 풀링(max pooling)** 과 **평균 풀링(average pooling)** 을 설명해주세요.
    - 적당히 패딩을 적용해 입력 이미지의 가장자리를 포함하고 모든 점을 중심으로 하는 P_ij를 만들 수 있고, 이 P_ij 내의 픽셀에 대해서 채널 k마다 독립적으로 H^2개 있는 픽셀값을 사용하여 하나의 픽셀값 u_ijk를 구할 수 있습니다.
    - 그 중에서 **최대 풀링**은 H^2개의 픽셀값 중 최대값을 고르는 방법, **평균 풀링**은 픽셀값의 평균을 계산하여 픽셀값으로 사용하는 방법을 의미합니다.

- Q. 풀링층의 특징을 설명해주세요.
    - 합성곱층과 마찬가지로 풀링층에도 2 이상의 스트라이드를 설정할 수 있으며 통상 이렇게 설정합니다.
    - 또한 합성곱층과 마찬가지로, 2층 신경망으로도 구성할 수 있고 층간의 결합이 국소적으로 제한되도록 구성된다.
        - **BUT** 이 경우 결합의 가중치는 합성곱층의 필터처럼 고정값으로 되어 있어 조절할 수 없다.
        - 따라서 풀링층에는 학습에 따라 변화할 수 있는 파라미터가 존재하지 않는다. 역전파법을 수행할 때 풀링층에는 델타에 대한 역전파 계산만을 수행한다.

- Q. 정규화층에 사용되는 방법과 **국소 콘트라스트 정규화(local contrast normalization)** 에 대해서 설명해주세요.
    - 명암을 여러 방법으로 정규화하는 방법 중에 이전에 설명해드렸던 것은, 이미지의 집합(훈련 데이터)에 대한 통계치를 이용하는 방법이 있었는데 정규화나 백색화가 있습니다.
    - CNN에서는 대상 이미지로부터 학습 이미지의 픽셀 단위 평균을 뺀 다음 CNN의 입력으로 사용하는 경우가 있습니다.
    - 이러한 방법 이외에 **국소 콘트라스트 정규화**가 있는데, 이 방법은 이미지 한장 한장에 대해 개별적으로 처리하는 방법입니다.[**국소 영역 내의 모든 채널의 픽셀을 대상으로 평균과 분산을 계산**]
        - CNN뿐만 아니라 일반적인 이미지 처리 방법 중 하나 입니다. 합성곱 연산이나 풀링과 마찬가지로 한 개 층으로 이 처리를 구현할 수 있고 역전파 계산도 가능합니다.

        - 하지만 풀링층과 마찬가지로 이층의 가중치는 고정값이어서 학습이 가능한 파라미터가 없습니다.
        - 국소 콘트라스트 정규화에는 **감산 정규화(subtractive normalization)** 와 **제산 정규화 (divisive normalization)** 두 가지가 있습니다.

- Q. 그러면 감산 정규화는 무엇인가요?

    - 감산 정규화는 입력 이미지의 각 픽셀 명암에서 P_ij(H x H 크기의 정사각형 모양의 영역)에 포함되는 **픽셀의 명암값의 평균**을 빼는 것을 의미합니다.
    - 이는 영역의 중앙부를 중요시하고 주변부의 영향력을 상대적으로 줄이기 위한 방법입니다. [영역의 중앙부에서 값이 최대가 되고 주변으로 갈수록 작아짐]

- Q. 제산 정규화는요?

    - 제산 정규화도 마찬가지로 국소 영역 내에서 이루어지지만 픽셀값의 분산을 추가적으로 필요로 합니다.
    - 감산 정규화를 거친 입력 이미지의 픽셀값을 다시 **분산의 제곱근**인 **표준편차**로 나누어줍니다.
    - 감산 정규화를 그대로 하게 되면 명암의 차이가 적은(대비가 적은) 국소 영역일수록 작은 명암의 차이가 증폭되어서 이미지의 노이즈가 강조되는 결과가 일어납니다.
        - 따라서 입력 이미지 내에서 대비가 큰 부분에만 적용되도록 처리를 해줍니다.