**Regularization**

높은 분산으로 신경망이 데이터를 과대적합(Overfitting)하는 문제가 의심된다면 가장 처음 시도해야할 것은 정규화(Regularization)이다. 높은 분산을 해결하기 위한 다른 방법으로 더 많은 훈련 데이터를 확보할 수도 있지만 비용이 많이 들어 가능하지 않은 방법일 수도 있다. 따라서 이럴 경우 정규화를 시도해볼 수 있다.  
  
정규화의 종류에는 L1 정규화와 L2 정규화가 있다. 모델을 압축하려는 목적이 있지 않는 한 대부분 사람들은 네트워크를 훈련할 때 L2 정규화를 많이 사용한다. 정규화를 하게 되면 손실함수에 추가적인 항을 더해주어 손실을 줄이기 위해 결국 가중치(W) 값들이 작아지게 된다.

결론적으로, 높은 분산 문제가 발생하였을 경우 더 많은 훈련 데이터 셋을 확보할 수 없다면 정규화를 시도해볼 수 있다.

**Why regularization reduces overfitting?**

과대적합(Overfitting)이 발생하는 원인에는 네트워크가 너무 커서 매우 복잡한 비선형 함수를 만들기 때문일 수도 있다. 즉, 다음과 같은 함수의 경우이다.  
텍스트, 친필, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명  
정규화는 이를 해결하여 과대적합의 가능성을 줄여줄 수 있다. 정규화를 할 경우 아래와 같이 추가적인 항(L2 정규화)을 더해준다.  
친필, 텍스트, 폰트, 서예이(가) 표시된 사진

자동 생성된 설명

이 때 정규화 매개변수 값을 크게 설정할 경우 손실(Cost)을 줄이기 위해 가중치 행렬(W)이 0에 상당히 가깝게 설정된다. 따라서 많은 은닉층의 유닛을 0에 가까운 값으로 설정하여 각 유닛들의 영향력을 줄이게 된다. 이는 직관적으로 보면 신경망이 훨씬 더 간단하고 작은 신경망의 역할을 하게 되는 것이다. 물론 여전히 층의 개수나 유닛의 수가 바뀐 것은 아니지만 역할은 로지스틱 회귀와 비슷해진다. 따라서 과대적합을 과소적합의 형태에 가깝게 만들어 줄 수 있다. 이 때 명심할 점은 가중치 매개변수를 적당한 값으로 정해야 한다.

왜 정규화가 과대적합을 막는데 도움을 주는지에 관한 또 다른 직관이 있다. 활성화 함수가 tanh일 경우를 생각해보자.  
라인, 도표, 그래프이(가) 표시된 사진

자동 생성된 설명  
즉, g(Z) = tanh(Z)인 경우이다. 이때 Z가 아주 작은 경우에는 아래의 그림과 같이 tanh 함수의 선형 영역에 위치하게 된다.  
라인, 도표, 그래프, 폰트이(가) 표시된 사진

자동 생성된 설명  
반대로 Z의 값이 커지면 선형 영역을 벗어난다. 위에서 보았듯이 정규화 매개변수가 커질 때 가중치(W)는 작아질 것이고, 가중치가 작아진다면 아래의 식에 의해 Z값 또한 작아질 것이다.  
친필, 폰트, 서예, 텍스트이(가) 표시된 사진

자동 생성된 설명  
따라서 Z가 상대적으로 작은 값을 갖게 된다면 g(Z) = tanh(Z)는 선형에 가까운 함수가 된다. 모든 층(Layer)이 선형이면 전체 네트워크 또한 선형이다. 따라서 매우 복잡한 비선형 함수를 만들 수 없다. 결국 높은 분산을 해결할 수 있게 된다.  
  
딥러닝에서 가끔 드롭아웃이라는 정규화 기법을 사용하기도 한다. L2나 L1 정규화 기법처럼 가중치가 너무 커지는 것을 막기위해 추가적인 항을 더해주는 것과 비슷한 효과를 볼 수 있다.

정리해보면, 정규화를 하게 되면 정규화 매개변수를 포함한 추가적인 항을 손실함수에 더하게 되고 그럼 결국 가중치(W)가 작아져 은닉층의 유닛들의 영향력이 작아지고 네트워크의 복잡도가 줄어든다. 또한 Z값이 작아져 tanh 활성화함수의 선형구간의 위치하여 선형함수를 만들어 과대적합을 막게 된다.

**Dropout Regularization**

L2 정규화 외에 또 다른 매우 강력한 정규화 기법에는 드롭아웃(Dropout)이 있다. 아래에 보이는 신경망은 과적합한 결과를 보인 신경망이라고 가정하자.  
라인, 도표, 원, 스케치이(가) 표시된 사진

자동 생성된 설명

드롭아웃(Dropout)이란 무작위로 신경망의 유닛을 삭제하는 기법이다. 드롭아웃을 사용하기 위해 먼저 신경망의 각각의 층에 대해 노드를 삭제할 확률을 설정한다. 그럼 각각의 층에 대해 각각의 노드마다 특정한 확률로 해당 노드를 유지하거나 삭제하게 된다. 예를 들어, 아래의 경우 각각의 노드마다 0.5의 확률로 해당 노드를 유지하고 0.5의 확률로 노드를 삭제하게 된다.  
라인, 도표, 원이(가) 표시된 사진

자동 생성된 설명

0.5(50%)의 확률상 다음과 같이 노드를 삭제했다고 하자.  
라인, 도표, 원, 디자인이(가) 표시된 사진

자동 생성된 설명

그럼 삭제된 노드의 들어가는 링크와 나가는 링크를 모두 삭제하고 결국 아래와 같은 간소화된 네트워크가 된다.  
  
라인, 도표, 원, 종이접기이(가) 표시된 사진

자동 생성된 설명

이 간소화된 네트워크에서 역전파도 진행된다. 노드를 무작위로 삭제하는 것이 이상한 기법처럼 보일 수도 있지만 실제로 잘 작동한다고 한다.

(역 드롭아웃을 구현하는 방법에 대한 설명은 유튜브 강의 영상 보면 확인할 수 있습니다.)

모델 훈련 후 테스트할 때 테스트 데이터 셋의 경우에는 드롭아웃을 사용하지 않는다. 왜냐하면 테스트를 할 때 결과가 무작위로 나오면 안되기 때문이다. 테스트 시에 드롭아웃을 할 경우 오히려 노이즈만 증가시킬 뿐이다. 여러 번 반복한 결과를 평균을 낼 수도 있지만 컴퓨터적으로 비효율적이며 결과도 결국 비슷하다고 한다.

**Understanding Dropout**

왜 드롭아웃(Dropout)이 정규화로 작동을 할까? 왜 정규화한 효과가 날까?  
첫 번째 직관은 더 작고 간소한 신경망을 사용하는 것과 같이 작동하여 정규화의 효과를 준다는 것이다.

두 번째 직관은 그림을 그려가며 이해를 해보자. 아래의 그림에는 입력(input)이 4개가 있고 오른쪽에 하나의 단일 유닛이 있다. 이 단일 유닛의 관점에서 살펴보자.  
친필, 원, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명  
이 단일 유닛이 하는 일은 입력을 받아 의미있는 출력을 생성하는 것이다. 드롭아웃을 통해 입력은 무작위(random)으로 삭제될 수 있다. 삭제할 확률이 0.5일때 아래의 두 가지(A와 B) 모두 가능한 경우이다.  
원, 도표, 라인, 디자인이(가) 표시된 사진

자동 생성된 설명  
A) 왼쪽 그림인 A의 경우는 input 2와 input 4만이 연결된다.  
원, 라인, 디자인이(가) 표시된 사진

자동 생성된 설명  
그리고 나머지 링크는 끊어지게 된다.  
원이(가) 표시된 사진

자동 생성된 설명  
B) 오른쪽 그림인 B의 경우는 input 1과 input 3만이 연결된다.  
원, 디자인이(가) 표시된 사진

자동 생성된 설명  
그리고 나머지 링크는 끊어지게 된다.  
원이(가) 표시된 사진

자동 생성된 설명  
이런 식으로 드롭아웃을 통해 무작위로 유닛들이 삭제되어 서로 다른 다양한 input을 오른쪽의 단일 유닛은 전달 받게 된다. 따라서 오른쪽의 단일 유닛은 어떠한 입력 특성에도 의존할 수 없다. 하나의 특정 입력에 유난히 큰 가중치를 부여할 수 없으므로 가중치를 4개의 input에 각각 분산시킨다. 즉, 가중치를 분산시킴으로써 가중치의 Norm(L2 정규화)이 줄어들게 된다. 따라서 드롭아웃은 L2 정규화와 같은 효과를 보여 과대적합을 막는데 도움이 된다.  
  
드롭아웃을 할 때 과대적합의 우려가 없는 층, 즉 유닛이 적어 많은 가중치(W)를 가지지 않는 층은 keep\_prob(층에서 유닛이 삭제되지 않을 확률)을 1로 주어 드롭아웃을 사용하지 않아도 된다. 반대로 과대적합의 우려가 높은 층, 즉 매개변수가 매우 많은 층은 더 강력한 형태의 드롭아웃을 위해 keep\_prob을 작게 설정하는 것이 좋다.

컴퓨터 비전 분야에서는 대부분의 경우 데이터가 부족하다. 따라서 이 분야에서는 드롭아웃이 매우 빈번하게 사용된다. 하지만 드롭아웃은 정규화 기법이고 과대적합을 막는데 도움을 주는 것으로 문제가 생기기 전까지는 사용하지 않는 것이 좋다. 단점이 존재하기 때문이다. 드롭아웃은 비용함수 또는 손실함수가 잘 정의되지 않는다는 단점이 존재한다. 무작위로 노드들이 삭제되므로 비용함수가 경사하강을 하는지 확인하는게 어렵다. 따라서 먼저 모델의 keep\_prob을 1로 설정하여 드롭아웃의 효과를 멈추고 비용함수가 단조 감소하는지 확인한다. 그 다음 문제가 있다면 드롭아웃의 효과를 다시 주고 코드는 바꾸지 않도록 한다.

결론적으로, 드롭아웃은 특정 입력 값에 의존하지 못하고 분산되어 가중치 값이 줄어들어 정규화 효과를 보이게 된다. 또한 모든 층에 똑같은 값의 keep\_prob을 설정하지 말고 매개변수가 많아 과대적합 우려가 큰 층은 낮은 keep\_prob을, 과대적합 우려가 작은 층은 높은 keep\_prob을 또는 아예 드롭아웃 효과를 끄기 위해 1로 설정하는 것이 좋다. 마지막으로 과대적합 문제가 발생하였을 경우에만 드롭아웃 기법을 사용하도록 한다.

**Other Regularization Methods**

L1, L2 정규화와 드롭아웃 정규화 기법 외에도 다음과 같은 정규화 기법들이 더 존재한다.

**Data augmentation**

전혀 다른 새로운 이미지를 확보하는 것이 물론 더 좋겠지만 비용이 많이 들기 때문에 이미지를 수평 방향으로 뒤집거나 확대하거나 회전하거나 왜곡 및 변형하여 샘플에 추가하는 방법도 있다.  
!텍스트이(가) 표시된 사진

자동 생성된 설명

이러한 추가적인 가짜 이미지들은 완전히 새로운 독립적인 고양이 샘플을 얻는 것보다 더 많은 정보를 추가해주지는 않을 것이다. 하지만 컴퓨터적인 비용이 들지 않고 할 수 있다는 장점이 있다.

**Early Stopping**

Early Stopping(조기 종료)라는 또 다른 기법이 있다. 경사 하강법을 실행하면서 훈련 데이터 셋 오차와 개발 데이터 셋 오차를 그래프에 그려보면 다음과 같은 그림이 나온다.  
텍스트, 라인, 도표, 그래프이(가) 표시된 사진

자동 생성된 설명

위의 그래프에서 발견할 수 있는 것은 개발 셋 오차함수 곡선이 감소하다가 어느 특정 부분에서 증가한다는 것이다. 조기 종료는 신경망이 이 지점에 위치할 때(개발 셋 오차함수가 증가하기 시작할 때) 작동한다. 중간에 신경망을 훈련시키는 것을 멈추고 이 개발 세트 오차를 만든 매개변수 값들을 최적으로 삼는다.  
이 방법이 정규화 효과를 보이는 이유는 처음에 무작위로 매우 작은 값으로 초기화한 가중치(W)는 반복을 실행할수록 커지는데 조기 종료 기법이 이를 멈추었기 때문이다. 멈춘 시점은 여전히 W가 그렇게 크지 않은 시점이다. 따라서 신경망을 덜 과대적합하게 만든다.  
  
조기 종료는 말 그대로 신경망을 조기에 종료시킨다. 하지만 이에는 단점이 존재한다. 비용함수를 최적화하는 것을 멈추게 하여 비용함수를 줄이는 일을 잘 하지 못하게 된다. 동시에 과대적합도 막으려하며 이는 문제를 더 복잡하게 한다. 따라서 조기 종료의 대안으로 L2 정규화를 사용할 수 있다. 그럼 가능한 오래 신경망을 훈련시킬 수 있게 된다. 단, L2 정규화에도 단점이 존재하는데 정규화 매개변수에 많은 값을 시도해야한다는 것이다. 반대로 조기 종료는 많은 값을 시도할 필요가 없다. 여러 정규화 기법에는 각각 장단점이 존재한다는 것을 알 수 있다.