

정규화

부산대학교 정보컴퓨터공학부

15 학번 이연걸

1. 개요

부적합이란 학습용 예제 자체를 제대로 풀지 못하는 현상을 뜻하고 과적합이란 전체적인 문제 특성을 파악하지 못한 채 학습 데이터 자체의 지역적 특성을 외워 버리는 바람에 나타나는 현상을 뜻한다.

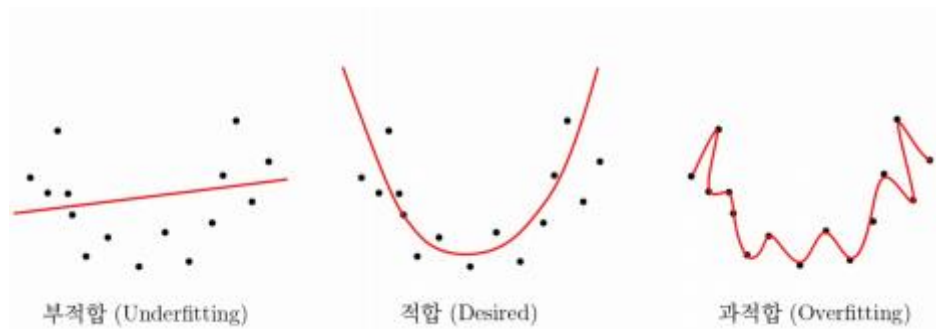


그림 1 – 부적합과 과적합

그림 1 을 보자. 점들은 학습되어야하는 데이터를 뜻하고 빨간 선은 학습된 모델이다. 부적합의 경우, 점들을 제대로 반영하지 못하였고 과적합의 경우, 점들을 지역적으로 받아들이는 모습이다.

부적합은 모델 용량을 키우고 학습 횟수를 늘리는 등 대처하기가 쉽지만 과적합은 이런 방법으로만으로는 해소가 어렵다. 이에 따라 학습 과정에 적당한 제약을 추가하여 학습용 예제 자체의 지역적 특성을 외워 버리기 어렵게 만드는 방법들이 고안되었다. 이 방법들이 바로 정규화 기법이다. 이 장에서는 L2 손실, L1 손실, 드롭아웃, 잡음 주입, 배치 정규화 등의 정규화 기법을 각각 적용한 뒤, 실험 결과를 비교해본다. 데이터는 꽃 이미지를 사용한다.

2. 구현 과정


파라미터에 대해 정규화를 진행하기 위하여 베이스라인 모델의 파라미터 분포를 확인한 뒤, L2 손실, L1 손실을 진행한다. 그 후, 계층에 대한 정규화를 진행하기 위하여 베이스라인 합성곱 신경망의 성능을 확인한 뒤, 드롭아웃, 잡음 주입, 배치 정규화 순으로 구현한다. 이미지 해상도는 [96, 96]으로 지정한다.

2.1 베이스라인 모델의 파라미터 분포 확인

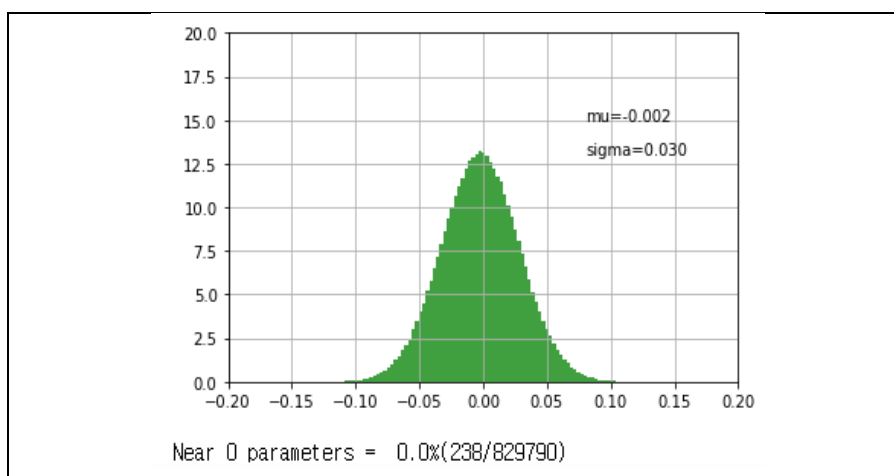
다음은 정규화 기법이 적용되지 않은 상태에서 파라미터들이 어떤 분포로 학습되는지 확인한 결과이다.

```
Model flowers_model_1 train started:
Epoch 2: cost=1.601, accuracy=0.246/0.290 (48/48 secs)
Epoch 4: cost=1.599, accuracy=0.246/0.210 (47/95 secs)
Epoch 6: cost=1.598, accuracy=0.246/0.300 (52/147 secs)
Epoch 8: cost=1.598, accuracy=0.246/0.260 (50/197 secs)
Epoch 10: cost=1.598, accuracy=0.246/0.210 (51/248 secs)
Model flowers_model_1 train ended in 248 secs:
Model flowers_model_1 test report: accuracy = 0.233, (0 secs)

Model flowers_model_1 Visualization
```



```
추정확률분포 [18,24,18,17,23] => 추정 dandelion : 정답 dandelion => 0
추정확률분포 [18,24,18,17,23] => 추정 dandelion : 정답 rose => X
추정확률분포 [18,24,18,17,23] => 추정 dandelion : 정답 tulip => X
```



2.2 L2 손실 및 L1 손실

L2 손실 및 L1 손실은 절대값이 큰 파라미터에 대해 불이익을 주어 값의 폭주를 막고 기왕이면 작은 절댓값의 파라미터들로 문제를 풀도록 압박하는 정규화 기법이다.

2.2.1 L2 손실

L2 손실 함수는 다음과 같다.

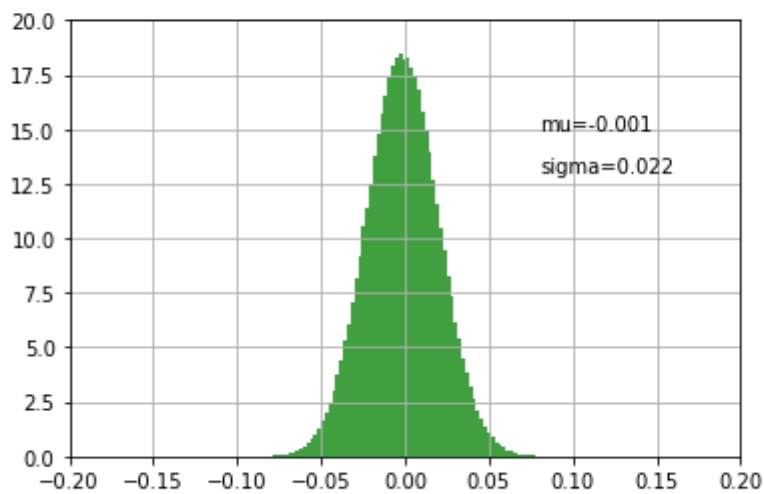
L2 손실함수

$$P_{L2} = \frac{1}{2} \lambda \sum_{i=1}^n w_i^2$$

$$L' = L + \frac{1}{2} \lambda \sum_{i=1}^n w_i^2$$

2.1의 구조에 L2 손실을 실험을 진행하였다. l2_decay 값은 0.1로 지정한다.

Model flowers_model_2 train ended in 983 secs:
Model flowers_model_2 test report: accuracy = 0.233, (0 secs)



Near 0 parameters = 0.0%(306/829790)

2.2.2 L1 손실

L1 손실 함수는 다음과 같다.

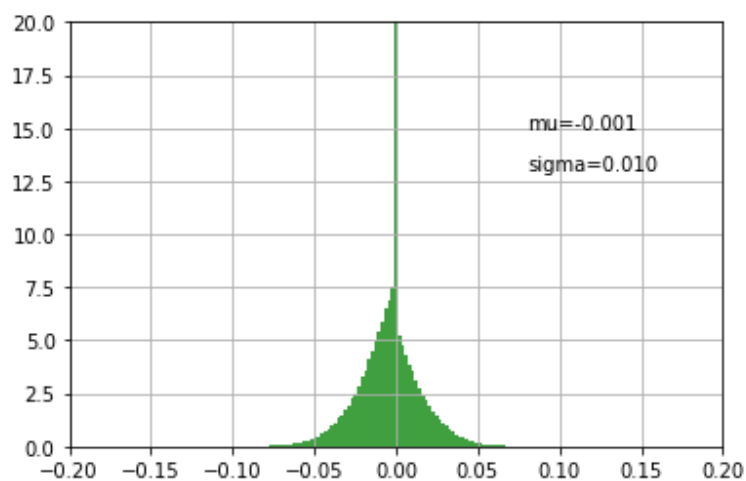
L1 손실함수

$$P_{L1} = \alpha \sum_{i=1}^n |w_i|$$

$$L' = L + \alpha \sum_{i=1}^n |w_i|$$

2.1 의 구조에 L1 손실을 실험을 진행하였다. l1_decay 값은 0.1 로 지정한다.

Model flowers_model_3 train ended in 1005 secs:
Model flowers_model_3 test report: accuracy = 0.233, (0 secs)




Near 0 parameters = 74.2%(615360/829790)

2.3 베이스라인 합성곱 신경망의 성능 확인

다음은 정규화 기법이 적용되지 않은 상태에서 파라미터들이 어떤 분포로 학습되는지 확인한 결과이다.

```
Model flowers_cnn_1 train started:
Epoch 2: cost=1.102, accuracy=0.557/0.690 (231/231 secs)
Epoch 4: cost=0.937, accuracy=0.657/0.550 (231/462 secs)
Epoch 6: cost=0.816, accuracy=0.694/0.680 (235/697 secs)
Epoch 8: cost=0.712, accuracy=0.738/0.690 (235/932 secs)
Epoch 10: cost=0.577, accuracy=0.785/0.610 (235/1167 secs)
Model flowers_cnn_1 train ended in 1167 secs:
Model flowers_cnn_1 test report: accuracy = 0.642, (6 secs)

Model flowers_cnn_1 Visualization
```



```
추정확률분포 [ 0, 0, 0.99, 0] => 추정 sunflower : 정답 sunflower => 0
추정확률분포 [ 0, 4, 23, 59, 13] => 추정 sunflower : 정답 tulip => X
추정확률분포 [ 0, 0, 0.100, 0] => 추정 sunflower : 정답 sunflower => 0
```

2.4 드롭아웃

드롭아웃 (dropout)이란 입력 또는 어떤 계층의 출력을 다음 계층에서 모두 이용하지 않고 일부만 이용하면서 신경망을 학습시키는 규제화 기법이다. 그림으로 살펴보면 다음과 같다.

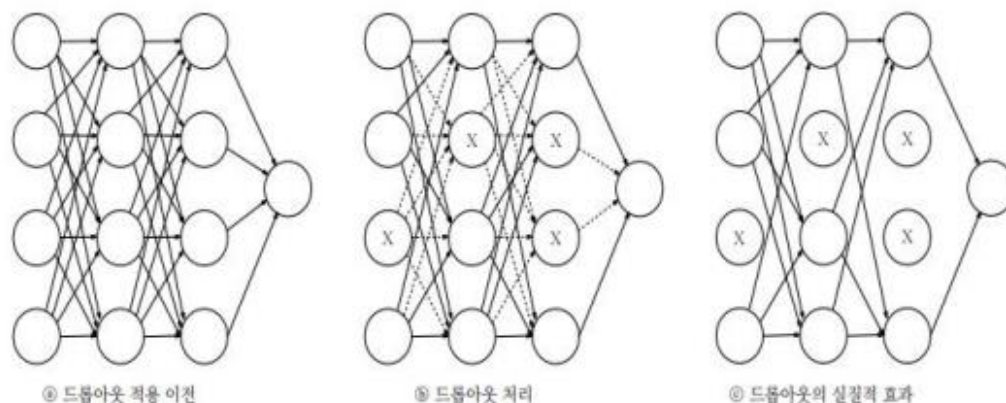


그림 2 – 드롭아웃

그림 ㉖와 ㉔에서 X 표시된 노드들은 드롭아웃 처리를 통해 배제되는 노드들을 나타낸다.

2.3 에서 합성곱 계층과 풀링 계층의 처리가 끝날 때마다 60%만 살리고 나머지 40%를 0 으로 만들어 걸러내는 드롭아웃 계층을 삽입한 결과 다음과 같다.

```
cnn2 = [['conv',{'ksize':3, 'chn':6}],
        ['max',{'stride':2}],
        ['dropout', {'keep_prob':0.6}],
        ['conv',{'ksize':3, 'chn':12}],
        ['max',{'stride':2}],
        ['dropout', {'keep_prob':0.6}],
        ['conv',{'ksize':3, 'chn':24}],
        ['avg',{'stride':3}],
        ['dropout', {'keep_prob':0.6}]]
fcnn2 = CnnRegModel('flowers_cnn_2', fd, cnn2)
fcnn2.exec_all(epoch_count=10, report=2, show_cnt=0)
```

```
Model flowers_cnn_2 train started:
Epoch 2: cost=1.279, accuracy=0.443/0.380 (244/244 secs)
Epoch 4: cost=1.131, accuracy=0.533/0.560 (238/482 secs)
Epoch 6: cost=1.020, accuracy=0.601/0.520 (266/748 secs)
Epoch 8: cost=0.949, accuracy=0.626/0.520 (345/1093 secs)
Epoch 10: cost=0.910, accuracy=0.659/0.540 (358/1451 secs)
Model flowers_cnn_2 train ended in 1451 secs:
Model flowers_cnn_2 test report: accuracy = 0.531, (17 secs)
```

2.5 잡음주입

잡음주입이란 두 계층 사이에 잡음 주입 계층을 삽입하고 이 계층이 아래 계층의 출력에 적당한 형태의 잡음을 추가하여 위 계층에 전달하는 정규화 기법이다. 잡음이 주입되면 위 계층의 학습은 그만큼 혼란을 겪게 되지만 그 결과 더욱 강건한 학습이 이루어져 다양한 입력의 변이를 더 잘 처리할 수 있게된다.

다음은 2.3 에서 각 합성곱 계층에 주어지는 입력에 평균 0, 표준편차 0.01 의 잡음을 추가하여 학습을 교란시킨 결과이다.

```
noise_std = 0.01
cnn3 = [['noise', {'type':'normal','mean':0,'std':noise_std}],
        ['conv',{'ksize':3, 'chn':6}],
        ['max',{'stride':2}],
        ['noise', {'type':'normal','mean':0,'std':noise_std}],
        ['conv',{'ksize':3, 'chn':12}],
        ['max',{'stride':2}],
        ['noise', {'type':'normal','mean':0,'std':noise_std}],
        ['conv',{'ksize':3, 'chn':24}],
        ['avg',{'stride':3}]]
fcnn3 = CnnRegModel('flowers_cnn_3', fd, cnn3)
fcnn3.exec_all(epoch_count=10, report=2, show_cnt=0)
```

```
Model flowers_cnn_3 train started:
Epoch 2: cost=1.085, accuracy=0.574/0.560 (389/389 secs)
Epoch 4: cost=0.863, accuracy=0.672/0.700 (414/803 secs)
Epoch 6: cost=0.759, accuracy=0.710/0.650 (354/1157 secs)
Epoch 8: cost=0.667, accuracy=0.752/0.630 (366/1523 secs)
Epoch 10: cost=0.569, accuracy=0.791/0.650 (348/1871 secs)
Model flowers_cnn_3 train ended in 1871 secs:
Model flowers_cnn_3 test report: accuracy = 0.624, (13 secs)
```

2.6 배치 정규화

배치 정규화 (batch normalization)이란 각 층의 활성화 함수의 출력값 분포가 골고루 분포되도록 강제하는 방법으로, 각 층에서의 활성화 함수 출력값이 정규분포 (normal distribution)를 이루도록 하는 방법이다. 학습하는 동안 이전 레이어에서의 가중치 매개변수가 변함에 따라 활성화 함수 출력값의 분포가 변화하는 내부 공변량 변화(Internal Covariate Shift) 문제를 줄여서 과적합을 억제한다.

그림으로 나타내면 다음과 같다.

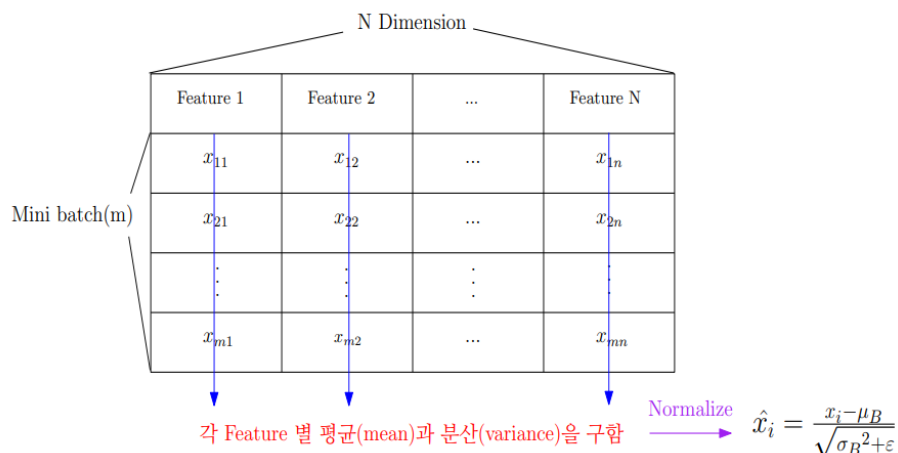


그림 3 - 배치 정규화

다음은 2.3 에서 각 합성곱 계층 앞에 배치 정규화 계층을 삽입한 결과이다.

```
cnn4 = [['batch_normal'],
        ['conv',{'ksize':3, 'chn':6}],
        ['max',{'stride':2}],
        ['batch_normal'],
        ['conv',{'ksize':3, 'chn':12}],
        ['max',{'stride':2}],
        ['batch_normal'],
        ['conv',{'ksize':3, 'chn':24}],
        ['avg',{'stride':3}]]
fcnn4 = CnnRegModel('flowers_cnn_4', fd, cnn4)
fcnn4.exec_all(epoch_count=10, report=2, show_cnt=0)
```

Model flowers_cnn_4 train started:

Epoch 2: cost=1.054, accuracy=0.592/0.440 (346/346 secs)

Epoch 4: cost=0.869, accuracy=0.664/0.240 (280/626 secs)

Epoch 6: cost=0.771, accuracy=0.708/0.320 (304/930 secs)

Epoch 8: cost=0.675, accuracy=0.745/0.410 (311/1241 secs)

Epoch 10: cost=0.593, accuracy=0.781/0.610 (317/1558 secs)

Model flowers_cnn_4 train ended in 1558 secs:

Model flowers_cnn_4 test report: accuracy = 0.556, (13 secs)

3. 총평

다음은 파라미터에 L2 손실과 L1 손실을 적용했을 때 나온 결과 표이다.

	기본	L2 손실	L1 손실
평균	-0.002	-0.001	-0.001
표준편차	0.03	0.022	0.010
정확도	23.3%	23.3%	23.3%
Near 0 parameters	0.0% (238)	0.0% (306)	74.2%

표 1 - 파라미터에 L2 손실과 L1 손실을 적용한 경우

L2 손실와 L1 손실을 적용한 경우, 정확도의 차이가 없는 점을 보아 학습에서는 차이가 없었다. 하지만, 평균과 표준편차가 줄어들었고, 이는 파라미터들의 절댓값이 크게 줄어들었음을 의미한다. 특히, L1 손실을 적용한 경우 전체 파라미터의 무려 74.2%가 0 에 가까운 값을 갖는 것으로 나타나 L1 손실이 확실한 효과를 발휘하고 있음을 잘 보여준다.

다음은 계층에 대해 드롭아웃, 잡음 주입, 배치 정규화를 적용했을 때 나온 결과 표이다.

	기본	드롭아웃	잡음 주입	배치 정규화
정확도	64.2%	53.1%	62.4%	55.6%

표 2 - 계층에 대해 드롭아웃, 잡음 주입, 배치 정규화를 적용한 경우

드롭아웃, 잡음 주입, 배치 정규화를 적용했지만 예상했던 것과 달리, 정확도가 낮아졌다. 정규화 기법은 학습 과정에 일부러 불리한 부담을 주는 것이기 때문에 이처럼 학습 성과가 떨어지는 것은 사실 당연하다.

더 많은 학습 횟수를 적용한다던가, 드롭아웃 적용 확률과 잡음 크기 등의 파라미터를 변경하면 더욱 나은 결과를 얻어낼 수 있을 것이다.