연구지도 및 연구윤리 12주차 보고서

(Chapter6. 매개변수 갱신 방법, 활성화 함수)

경북대학교 전자공학부
2016113566 김남영

1. 매개변수 갱신 방법

1) SGD(확률적 경사 하강법)

손실 함수를 최소화 하기 위하여 기울기를 이용하는 방법이다. 즉 기울어진 방향으로 일정 거리(학습률만큼)를 이동한다. 이를 경사 하강법이라 하는데, 이때 데이터를 미니배치로 무작위로 선정하기 때문에 확률적 경사 하강법이라 부른다. 기존의 경사 하강법을 사용하면 학습률에 따라 local minima(국소적인 범위에서 최솟값)에 빠지는 경우도 있지만, 확률적 경사 하강법은 미니배치를 활용하여 여러 번 최적의 갱신 값을 찾기 때문에 local minima에서 탈출하기 쉽다. 하지만 기울기가 가리키는 방향이 목표 지점(최솟값)이 아닌 경우 탐색 경로가 비효율적이다.

$$W \quad W - \eta \frac{\partial L}{\partial W}$$

2) Momentum(모멘텀)

기존의 경사 하강법에 관성의 원리를 추가한 갱신 방법이다. 즉, 과거에 이동했던 방향을 새롭게 계산되는 기울기 값에 반영하여 추가적으로 이동한다. 기존에 이동하던 방향을 기억하므로 local minima를 쉽게 빠져나가는 효과를 기대할 수 있으며, 특히 SGD가 진동(Oscilation)하고 있을 때에도 목표 지점으로 가려는 관성에 의해 진동이 억제되어 더 빠르게 목표 지점에 도달할 수 있다.

$$v \leftarrow \alpha v - \eta \frac{\partial L}{\partial W}$$
$$W \leftarrow W + v$$

3) Nesterov Momentum

Momentum optimizer가 관성에 의해 목표지점을 추월하는 현상이 발생하므로 이전의 가속도를 줄여서 반영하고자 고안된 방법이다. Momentum은 이전의 관성을 현재 기울기 값에 더하여 구하지만 Nesterov Momentum은 관성에 의한 이동을 먼저 반영하고 그 상태에서 기울기 값을 더하여 추월하는 현상을 줄인다.

4) AdaGrad

매개 변수를 갱신할 때 매개 변수마다 학습률을 다르게 설정하는 갱신 방법이다. h라는 변수에 기존 기울기 값을 제곱하여 더해주고, 매개 변수를 갱신할 때 학습률에 1/sqrt(h)를 곱하여 학습률을 조정한다. 매개 변수의 각 원소 중 갱신이 많이 된(기울기가 큰) 원소는 h값이 크기 때문에 학습률이 낮아진다. 갱신이 크게 될수록 목표 값에 가까이 있을 확률이 높기 때문에 세밀하게 값을 조정하는 것이다. 하지만 h값은 학습을 진행할수록 누적되어 커지므로 학습률은 계속해서 낮아진다. 따라서 어느 순간 갱신량이 0에 가까워져 갱신이 되지 않는 문제가 발생한다.

$$h \quad h + \frac{\partial L}{\partial W} \odot \frac{\partial L}{\partial W}$$
$$W \leftarrow W - \eta \frac{1}{\sqrt{h}} \frac{\partial L}{\partial W}$$

5) RMSProp

AdaGrad의 단점(갱신량이 0에 가까워짐)을 극복하기 위해 제안된 방법이다. AdaGrad의 경우 h값을 구할 때 기울기 값의 원소별 제곱을 이용하지만 RMSProp의 경우 지수평균을 사용하여 h가 무한히 커지지 않는다. 다음과 같이 ρ를 추가하여 h는 무한히 커지지 않으며 ρ가 작을수록 최신의 기울기 값을 더 크게 반영한다.

$$h_i \leftarrow \rho h_{i-1} + (1-\rho) \frac{\partial L_i}{\partial W} \odot \frac{\partial L_i}{\partial W}$$

6) Adam

RMSProp과 Momentum의 장점을 합친 최적화 방법이다. 현재의 기울기 값에 이전의 기울기 값을 누적하여 관성의 효과를 반영하고(Momentum), 현재 기울기 값의 원소별 제곱으로 학습률을 조정하여 목표 지점에 가까울수록 세밀한 조정이 가능하게끔 하고, 최신의 기울기 값을 크게 반영하도록 만든다(RMSProp).

7) Nadam

Adam이 RMSProp와 Momentum이 결합된 방법이라면 Nadam은 Momentum대신 Nesterov Momentum을 활용한 방법이다.

2. 딥러닝에서 사용하는 활성화 함수

1) identity function

입력을 그대로 출력하는 함수

2) step function

입력이 0보다 크면 1을 출력하고 0보다 작으면 0을 출력하는 함수

3) sgn function

입력이 0보다 크면 1을 출력하고 0보다 작으면 -1을 출력하는 함수

4) sigmoid function

1 / (1 + exp(-x)) 값을 출력하는 함수

5) tanh function

2 / (1 + exp(-2*x)) - 1 을 출력하는 함수

6) softsign function

x / (1 + x의 절댓값) 을 출력하는 함수

7) arctan function

$\tan^{-1}x$를 출력하는 함수

8) Relu function

입력이 0보다 작으면 0을 출력하고 0보다 크면 입력을 그대로 출력하는 함수

9) elu function

Relu function의 음에 값에 exponential 값을 적용한 함수(Relu는 값이 음수일 때 학습을 하지 못하므로)

10) threshold relu function

relu function의 임계값을 조정가능하게 만든 함수

11) softplus function

log(1 + exp(x))를 출력하는 함수

12) bentidentity function

(sqrt(x*x+1)-1)/2+x를 출력하는 함수

13) gaussian function

exp(-x*x)를 출력하는 함수

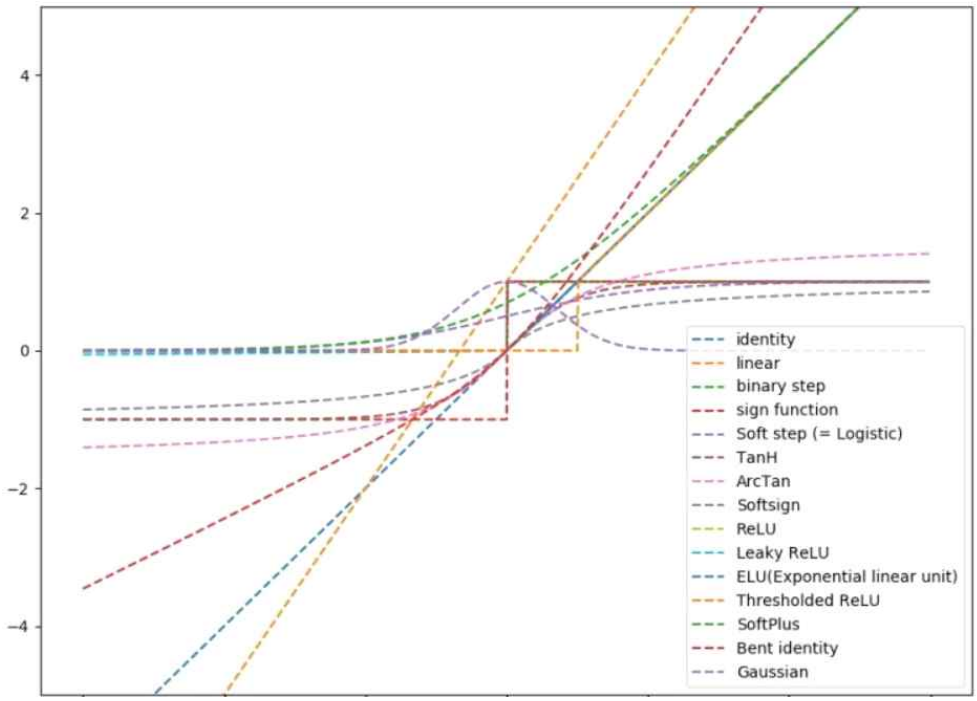

Fig. 1. 위 함수들의 출력 예시

3. 가중치 초깃값의 표준편차

11주차 과제 진행시 5층 신경망에 입력을 넣었을 때 다음과 같이 손실 함수의 갱신이 이루어지지 않는 문제가 발생하였다.

```
coding/spyder/ch3')
Reloaded modules: dataset.mnist, five_layer_net
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.10441666666666667 0.1028
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
```

chapter 6를 공부하면서 표준편차를 0.01로 설정 했을 때 각 층의 활성화 값이 아주 작은 값을 가진다는 것을 알 수 있었다. 작은 값의 데이터가 다음 층으로 전달되므로 역전파 시에도 기울기가 크게 작아지고, 이는 갱신이 거의 되지 않는다는 것을 의미한다. 따라서 책에서 제시한 He 초깃값을 적용하여 다시 한 번 실행해보았다.

```python
def __init__(self, input_size, hidden1_size, hidden2_size, hidden3_size, hidden4_size, output_size):
    # 가중치 초기화
    weight_init_std = np.sqrt(2/hidden1_size)
    self.params = {}
    self.params['W1'] = weight_init_std * np.random.randn(input_size, hidden1_size)
    self.params['b1'] = np.zeros(hidden1_size)
    self.params['W2'] = weight_init_std * np.random.randn(hidden1_size, hidden2_size)
    self.params['b2'] = np.zeros(hidden2_size)
    self.params['W3'] = weight_init_std * np.random.randn(hidden2_size, hidden3_size)
    self.params['b3'] = np.zeros(hidden3_size)
    self.params['W4'] = weight_init_std * np.random.randn(hidden3_size, hidden4_size)
    self.params['b4'] = np.zeros(hidden4_size)
    self.params['W5'] = weight_init_std * np.random.randn(hidden4_size, output_size)
    self.params['b5'] = np.zeros(output_size)
```

weight_init_std 값을 He 초깃값으로 수정.

```
0.12055 0.1157
0.8010833333333334 0.8104
0.8558833333333333 0.861
0.8818833333333334 0.8847
0.8953666666666666 0.8964
0.9051166666666667 0.9082
0.9127833333333333 0.9142
0.9185666666666666 0.9188
0.9264833333333333 0.9265
0.9287333333333333 0.9275
0.9323166666666667 0.9313
0.9363833333333333 0.9354
0.9375666666666667 0.9345
0.94175 0.939
0.9435 0.9385
0.9438166666666666 0.9388
0.9461833333333334 0.942
```

He 초깃값으로 수정 후 5층 신경망의 실행 결과. 손실함수의 갱신이 정상적으로 이루어짐.

4. 고찰

1) 저번 주에 해결되지 않았던 문제(매개 변수가 갱신되지 않는 문제)가 바로 해결되어 다행이다.
2) 지금은 Xavier 초깃값, He 초깃값에 대해 간략하게 설명되어 있지만, 수학적으로 파고든다면 아주 어려울 것 같다.
3) 책에서 히스토그램을 이용해서 활성화 값이 어떻게 분포하고, 학습이 제대로 이루어 지고 있는지를 확인 했는데, 앞으로도 이 방법을 활용하면 코드를 어떻게 수정해야할지에 대해 큰 도움을 얻을 수 있을 것 같다.
4) 생각보다 아주 많은 활성화 함수가 존재한다는 것을 알게 되었다. 상황마다 활성화 함수를 선정하는 방법이 궁금하다.