ch17_8_back_propagation

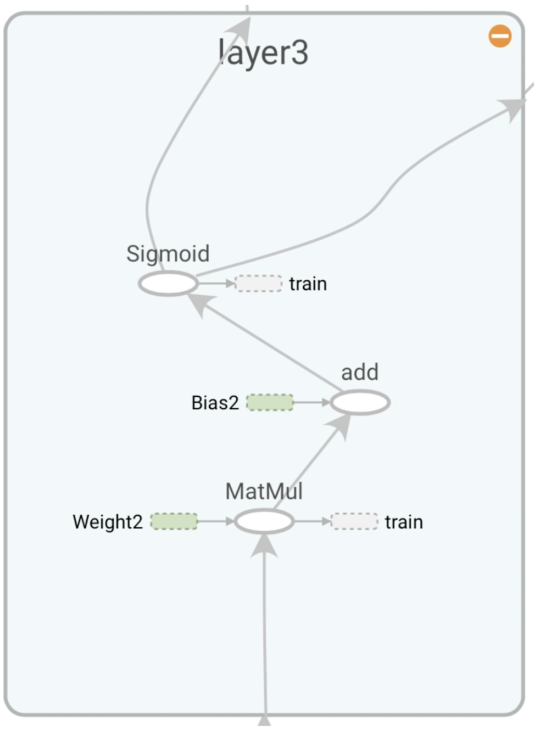
Tags

Back Propagation

이전 챕터에서 chain rule에 대해서 살펴봤습니다. Back Propagation 알고리즘은 chain rule 을 이용하여 Multilayer Perceptron 을 학습시키는 모델입니다.

앞서 민스키 교수가 자신의 저서에서 지구상의 누구도 MLP를 학습시키는 방법을 발견하지 못했다고 못박은 것을 기억하실 겁니다. 이후에 오류 역전파 기법과 웨이트 초기화, 비선형 활성화 함수의 발견 등으로 MLP 학습이 가능해지면서 지금의 AI 시대에 이르게 되었습니다. 그렇다면 Back Propagation 알고리즘에 대해서 살펴보겠습니다.

Computational Graph

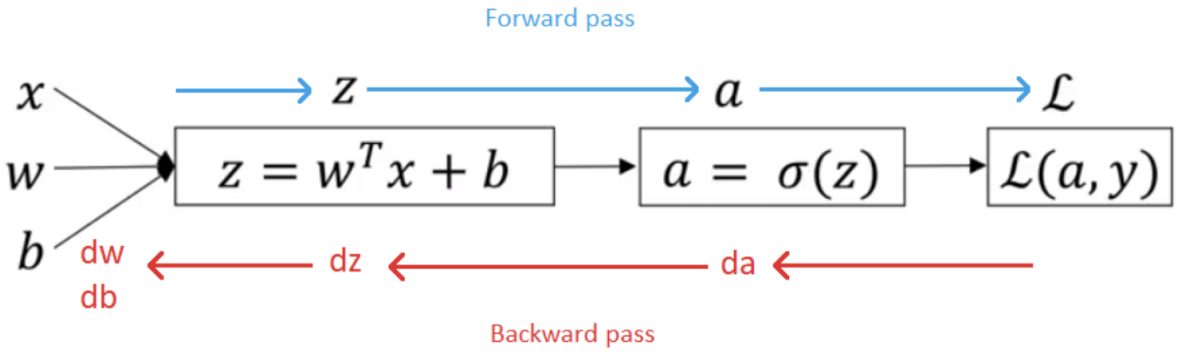


Back propagation in
TensorFlow
TensorBoard

```
hypothesis = tf.sigmoid(tf.matmul(L2, W2) + b2)
```

먼저 뉴럴 네트워크를 복잡한 형태의 합성 함수로 여기고 그래프 형태로 그려줍니다. 이를 Computational Graph라고 부릅니다.

Forward Propagation



뉴럴 네트워크를 이용해서 예측 값을 계산합니다. 이를 forward propagation이라고 부릅니다. 예측 값을 계산했으면 이제 loss function을 이용해서 loss를 계산합니다.

Back Propagation

이제 뉴럴 네트워크를 구성하는 각각의 weight들이 최종 결과값에 얼만큼 영향을 미쳤는 지를 chain rule을 이용해서 계산합니다. 영향을 많이 미친 weight는 크게 조정하고, 영향을 작게 미친 weight는 작게 조정하면 되겠죠? 각 weight들의 편미분 값과 learning rate를 곱한 값만큼 조정해주면 됩니다.

$$w = w - \alpha \frac{\partial L}{\partial w} \ (\alpha : \text{learning rate})$$

이 과정을 많은 데이터 셋에 대해서 반복하면 multilayer perceptron 모델을 학습시킬 수 있습니다. 오류 역전파 과정을 직접 계산해보고 싶으시다면 아래 문서를 참고하시면 좋습니다.

https://wikidocs.net/37406

이러한 역전파 알고리즘을 직접 구현하는 것은 굉장히 까다롭습니다. 때문에 torch나 tensorflow에 내장되어 있는 구현체를 사용하며, 이전 실습때 사용하였던 SGDoptimizer도 이 중 하나입니다. 이 외에도 역전파 알고리즘을 약간씩 개선한 optimizer들이 등장했고, 현재도 연구가 이루어지는 분야입니다. 그 중에서도 AdamOptimizer를 많이 사용하며, 다음 실습때 사용할 예정입니다.

다양한 optimizer들에 대해서 궁금하시다면 아래 영상을 참고하시면 좋습니다. (난이도가 무시무시합니다.)

https://www.youtube.com/watch?v=KN120w3PZIA

정리

딥러닝 모델을 학습시키는 핵심 알고리즘인 오류 역전파를 정리하면 다음과 같습니다. 이 과정을 잘 이해하시고 다음 챕터로 넘어가겠습니다.

1. 모델을 computational graph로 표현한다.
2. 순전파를 통해 예측값을 계산한다.
3. loss function을 이용하여 loss를 계산한다.
4. chain rule을 이용하여 모델의 각 weight 별로 loss에 대한 편미분 값을 계산한다.
5. 편미분 값과 learning rate를 가지고 기존 weight를 업데이트한다.
6. 이 과정을 반복하여 loss를 줄이는 방향으로 모델을 학습시킨다.