밑바닥 딥러닝1 - chapter 6

chapter6 - 학습 관련 기술들

6.1 매개변수 갱신

- [최적화] : 신경망 학습의 목적은 손실 함수의 값을 가능한 한 낮추는 매개변수를 찾는 것
- [확률적 경사 하강법(SGD)]

6.1.3 SGD의 단점

- SGD의 이러한 단점을 개선해주는 모멘텀, AdaGrad, Adam이라는 세 방법이 있다.

6.1.4 모멘텀

- v라는 변수가 새로 나오는데, 이는 물리에서 말하는 속도
- 모멘텀은 공이 그릇 바닥을 구르듯 움직입니다.
- SGD와 비교하면 '지그재그 정도'가 덜한 것을 알 수 있죠
- x축의 힘은 아주 작지만 방향은 변하지 않아서 한 방향으로 일정하게 가속하기 때문이다.

6.1.5 AdaGrad

- 학습률을 서서히 낮추는 가장 간단한 방법은 매개변수 '전체'의 학습률 값을 일괄적으로 낮추는 것이겠죠.
- AdaGrad는 '각각의' 매개변수에 '맞춤형' 값을 만들어줍니다.
- h는 [식 6.5]에서 보듯 기존 기울기 값을 제곱하여 계속 더해줍니다. 그리고 매개변수를 갱신할 때 1/sqrt(h)를 곱해 학습률을 조정합니다.
- 그림 [6-6]을 보면 최솟값을 향해 효율적으로 움직이는 것을 알 수 있습니다. y축 방향은 기울기가 커서 처음에는 크게 움직이지만, 그 큰 움직임에 비례해 갱신 정도도 큰 폭으로 작아지도록 조정됩니다. 그래서 y축 방향으로 갱신 강도가 약해졌고, 지그재그 움직임이 줄어듭니다.

6.1.6 Adam

- 모멘텀은 공이 그릇을 구르는 듯한 물리 법칙
- AdaGrad는 매개변수의 원소마다 적응적으로 갱신 정도를 조정했습니다.
- 이 두 기법을 융합하면 Adam이 탄생!
- 하이퍼파라미터의 '편향 보정'이 진행된다는 점도 Adam의 특징입니다.

6.1.7 어느 갱신 방법을 이용할 것인가?

- SGD, 모멘텀, AdaGrad, Adam의 네 후보 중 어느 것을 채택하면 될까요? 유감스럽게도 모든 문제에서 항상 뛰어난 기법이라는 것은 (아직까진) 없습니다.
- 지금도 많은 연구에서 SGD를 사용하고 있습니다. 모멘텀과 AdaGrad도 시도해볼 만한 가치 있는

방법입니다. 요즘에는 많은 분들이 Adam에 만족해하며 쓰는 것 같습니다.

6.2 가중치의 초깃값

6.2.1 초깃값을 0으로 하면?

- [가중치 감소] : 오버피틴ㅇ을 억제해 범용 성능을 높이는 테크닉
- 가중치 감소는 간단히 말하자면 가중치 매개변수의 값이 작아지도록 학습하는 방법입니다. 가중치 값을 작게 하여 오버피팅이 일어나지 않게 하는 것이죠.
- 지금까지 가중치의 초깃값은 0.01 * np.random.randn(10, 100)처럼 정규분포에서 생성되는 값을 0.01배 한 작ㅇ느 값(표준편차가 0.01인 정규분포)을 사용했습니다.
- 초깃값을 모두 0으로 해서는 안 되는 이유는 뭘까요? 그 이유는 바로 오차역전파법에서 모든 가중치의 값이 똑같이 갱신되기 때문입니다.

6.2.2 은닉층의 활성화값 분포

- [기울기 소실] : 층을 깊게 하는 딥러닝에서는 기울기 소실은 더 심각한 문제가 될 수 있습니다. (값이 특정 값에 극단저긍로 분포되는 현상)
- 앞의 예처럼 0과 1로 치우치진 않았으니 기울기 소실 문제는 일어나지 않습니다만, 활성화값들이 치우쳤따는 것은 표현력 관점에서는 큰 문제가 있는 것입니다.
- 이 상황에서는 다수의 뉴런이 거의 같은 값을 출력하고 있으니 뉴런을 여러 개 둔 의미가 없어진다는 뜻입니다.
- 그래서 활성화값들이 치우치면 [표현력을 제한]한다는 관점에서 문제가 됩니다.
- [Xaiver 초깃값] : 활성화 함수가 선형인 것을 전제 – sigmoid 함수, tanh 함수
- 이 결과를 보면 xaiver 초깃값을 이용함으로써 층이 깊어지면서 형태가 다소 일그러지지만, 앞에서 본 방식보다는 확실히 넓게 분포됨을 알 수 있습니다. 각 층에 흐르는 데이터는 적당히 퍼져 있으므로, 시그모이드 함수의 표현력도 제한받지 않고 학습이 효율적으로 이뤄질 것으로 기대됩니다.

6.2.3 ReLU를 사용할 때의 가중치 초깃값

- 활성화 함수 ReLU에 적합한 초깃값 [He 초깃값]
- 다시 정리하면, 활성화 함수로 ReLU를 사용할 때는 He 초깃값을, sigmoid나 tanh 등의 S자 모양 곡선일 때는 Xavier 초깃값을 씁니다.

6.3 배치 정규화

- 가중치의 초깃값을 적절히 설정하면 각 층의 활성화값 분포가 적당히 퍼지면서 학습이 원활하게 수행된다는 것을 배웠습니다.
- 그렇다면 각 층이 활성화를 적당히 퍼뜨리도록 '강제'해보면 어떨까요? 실은 [배치 정규화]가 그런 아이디어에서 출발한 방법입니다.

6.3.1 배치 정규화 알고리즘

- [배치 정규화]가 주목받는 이유는 다음과 같습니다.

1. 학습을 빨리 진행할 수 있다. (학습 속도 개선)
2. 초깃값에 크게 의존하지 않는다(골치 아픈 초깃값 선택 장애여 안녕!)
3. 오버피팅을 억제한다(드롭 아웃 등의 필요성 감소)

- [배치 정규화] : 각 층에서의 활성화값이 적당히 분포되도록 조정하는 것
- 그래서 [배치 정규화 계층]을 신경망에 삽입합니다.
- Affine – Batch Norm – Relu / – Affine – Batch Norm – ReLU / – Affine – Softmax

6.3.2 배치 정규화의 효과

- 거의 모든 경우에서 배치 정규화를 사용할 때의 학습 진도가 빠른 것으로 나타납니다. 실제로 배치 정규화를 이용하지 않는 경우엔 초깃값이 잘 분포되어 있지 않으면 학습이 전혀 진행되지 않는 모습도 확인할 수 있습니다.

6.4 바른 학습을 위해

- 기계학습에서는 [오버피팅]이 문제가 되는 일이 많습니다.

6.4.1 오버피팅

- [오버피팅]은 1) 매개변수가 많고 표현력이 높은 모델 2) 훈련 데이터가 적기 때문에 발생합니다.

6.4.2 가중치 감소

- [오버피팅] 억제용으로 예로부터 많이 이용해온 방법 중 [감중치 감소]라는 것이 있습니다.
- 이는 학습 과정에서 큰 가중치에 대해서는 그에 상응하는 큰 페널티를 부과하여 오버피팅을 억제하는 방법입니다.
- 신경망 학습의 목적은 손실 함수의 값을 줄이는 것입니다. 이때, 예를 들어 가중치의 제곱 법칙(L2 법칙)을 손실 함수에 더합니다.

6.4.3 드롭아웃

- 오버피팅을 억제하는 방식으로 손실 함수에 가중치의 L2 법칙을 더한 가중치 감소 방법을 설명했습니다. 가중치 감소는 간단하게 구현할 수 있고 어느 정도 지나친 학습을 억제할 수 있습니다. 그러나 신경망 모델이 복잡해지면 가중치 감소만으로는 대응하기 어려워집니다.
- 그래서 [드롭 아웃]을 이용합니다. 드롭아웃은 뉴런을 임의로 삭제하면서 학습하는 방법입니다. 훈련 때는 데이터를 흘릴 때마다 삭제할 뉴런을 무작위로 선택하고, 시험 때는 모든 모든 뉴런에 신호를 전달합니다. 단, 시험 때는 각 뉴런의 출력에 훈련 때 삭제한 비율을 곱하여 출력합니다.
- 순전파를 담당하는 forward 메서드에서는 훈련 때 (train_flg = True일 때)만 잘 계산해두면 시험

때는 단순히 데이터를 흘리김나 하면 됩니다. 삭제한 비율은 곱하지 않아도 좋습니다. 실제 딥러닝 프레임워크들도 비율을 곱하지 않습니다.

6.5 적절한 하이퍼파라미터 값 찾기

6.5.1 검증 데이터

- [훈련 데이터]로는 학습을 하고, [시험 데이터]로는 범용 성능을 평가합니다.
- 하이퍼파라미터를 조정할 때는 하이퍼파라미터 전용 확인 데이터가 필요합니다. 이를 [검증 데이터]라고 합니다.
- [훈련 데이터]는 매개변수의 학습에 이용하고, 검증 데이터는 하이퍼파라미터의 성능을 평가하는 데 이용합니다. 시험 데이터는 범용 성능을 확인하기 위해서 마지막에 (이상적으로는 한 번만) 이용합니다.
- MNIST 데이터셋에서 검증 데이터를 얻는 가장 간단한 방법은 훈련 데이터 중 20% 정도를 검증 데이터로 먼저 분리하는 것입니다.

6.5.2 하이퍼파라미터 최적화

- 하이퍼파라미터를 최적화할 때의 핵심은 하이퍼파라미터의 '최적 값'이 존재하는 범위를 조금씩 줄여간다는 것입니다. 우선 대략적인 범위를 설정하고 그 범위에서 무작위로 하이퍼파라미터 값을 골라낸(샘플링) 후, 그 값으로 정확도를 평가합니다.
- [0단계] : 하이퍼파라미터의 값의 범위를 설정한다. - [1단계] : 설정된 범위에서 하이퍼파라미터의 값을 무작위로 추출한다. - [2단계] : 1단계에서 샘플링한 하이퍼파라미터 값을 사용하여 학습하고, 검증 데이터로 정확도를 평가한다.(단, 에폭은 작게 설정한다.) - [3단계] 1단계와 2단계를 특정 횟수(100회 등) 반복하며, 그 정확도의 결과를 보고 하이퍼파라미터의 범위를 좁힌다.