**DL-HW #05**

2015004693\_양상헌

*실행환경: MAC OS TERMIINAL( MACBOOK PRO 2015 RETINA , MOJAVE 10.14.6 ) , ANACONDA*

1. **Source Code:**

* *Assignment5 폴더 참조 (DL\_HW\_05week.py & DL\_HW\_04week.py)*

“DL\_HW\_04week.py” 는 Dropout 을 적용하지 않고, 기본적인 Gradient Descent Optimizer를 적용했던 4주차 과제 파일이고, “DL\_HW\_05week.py” 는 이번주 과제인 Dropout 과 ADAM Optimizer를 “DL\_HW\_04week.py”에 적용한 것이다.(Dropout Probability 는 70% 로 설정하였고, ADAM Optimizer에서 beta1=0.6, beta2=0.8, epsilon=0.1, learning\_rate=0.9 로 설정하였다.) 결과 비교를 위해 두개의 파일을 모두 추가하였다.

1. **Result & Comparison:**

**\*NO-Dropout & Gradient Descent Optimizer (HW04)**

*(출력물 중간 생략)*

스크린샷, 앉아있는이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명 (1)

스크린샷, 앉아있는이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명 (2)

스크린샷, 앉아있는이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명 (3)

스크린샷, 앉아있는이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명 (4)

스크린샷, 앉아있는이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명 (5)

Overall :

(1) ~ (5) 총 5번 실행 후 결과를 확인해보면 대체적으로 거의 비슷한 정도의 정확도(약 96.5%)가 나오는 것을 확인할 수 있다. 또한 1Epoch이 진행될 때마다 변화하는 Cost의 양이 일정하게 점진적으로 줄어드는 경향을 확인 할 수 있어 안정적으로 학습이 잘 이뤄진다고 볼 수 있을 것 같다.

**\*Dropout & ADAM Optimizer (HW05)**

*(출력물 중간 생략)*

스크린샷, 앉아있는이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명(1)

스크린샷, 앉아있는이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명(2)

스크린샷, 앉아있는이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명(3)

스크린샷, 앉아있는이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명(4)

스크린샷, 앉아있는이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명(5)

스크린샷, 앉아있는이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명(6)

스크린샷, 앉아있는이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명(7)

스크린샷, 건물, 병, 앉아있는이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명(8)

스크린샷, 앉아있는이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명(9)

스크린샷, 건물, 병, 앉아있는이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명(10)

Overall :

(1)~(10) 총 10번 실행 후 결과를 확인해 보면 앞서 실행해봤던 No-Dropout & GD Optimizer일 때와는 다른 경향의 결과가 나오는 것을 확인할 수 있다. 총 10번의 실행에서 90% 이상의 정확도가 나오는 경우는 총 6번( (1)-0.9483, (2)-0.939, (5)-0.9282, (7)-0.9461, (8)-0.9477, (10)-0.9479 )밖에 되지 않는다. 그 외에 4번( (3)-0.2967, (4)-0.1028, (6)-0.4329, (9)-0.2061 )의 결과에서는 10% 1회, 20% 1회, 30% 1회, 43% 1회의 결과를 확인할 수 있었는데, 앞선 No-Dropout & GD Optimizer일 때보다 정확도의 측면에서 좋지 않은 결과가 나온다고 할 수 있다. 또한 1Epoch이 진행될 때마다 변화하는 Cost의 양이 일정하게 점진적으로 줄어들지 않을 때가 앞선 결과에 비해 덜 안정적으로 학습이 이뤄진다고 볼 수 있을 것 같다.

1. **Disscussion:**

**< 코드 설명 >**

코드의 전체적인 틀은 4주차 과제와 동일한 구조를 가지고, DROPOUT과 ADAM Optimizer가 추가된 형태이다. 4주차 과제와 마찬가지로 전체 트레이닝 데이터를 총 15번 학습하게 되고, 전체 트레이닝 데이터가 1번 학습될 때의 batch의 사이즈는 100으로 하여 전체 데이터 숫자인 55000 에 100을 나눈 결과인 550만큼 for문을 반복하며 학습하게 된다. 또한 15번 학습을 반복할 때, 전체 트레이닝 데이터에 대한 평균 cost를 출력하도록 하고, 마지막에 최종 test 데이터로 모델의 정확성을 확인하여 이를 수치상으로 출력하도록 해주었다.

또한 MLP의 구조는 각각 784->256->128->256->10개의 뉴런으로 구성되고 총 4개의 Layer로 이뤄져있는 네트워크의 모델로 디자인 하였고, 마지막 Layer를 제외한 나머지 Layer에서는 Activation Function으로 Sigmoid를 사용하고, 마지막 Layer에서는 Softmax를 Activation Function으로 사용하였다. 이때 각 Layer마다 Dropout기능을 추가해 주었는데 Dropout 의 확률은 70%로 전체 뉴런 중에서 랜덤한 70%의 뉴런만 active하게 된다.

ADAM Optimizer를 사용할 때는 Beta1=0.6, beta2=0.8, epsilon=0.1, learning\_rate=0.9 로 설정하여 주었는데, 이 설정 값은 여러 번 변경을 해본 후 결과가 가장 잘 나온다고 생각한 숫자를 대입하여 사용한 것이다.

**< 결과 분석 >**

앞서 확인한 결과들을 비교해 봤을 때 DROPOUT을 적용하지 않고 Gradient Descent Optimizer를 사용했을 때 DROPOUT을 적용하고 ADAM Optimizer를 사용 했을 때보다 더 높은 정확성을 보이게 된다.

이러한 이유로 생각되는 부분은 먼저 Optimizer가 얼마나 빠르고 정확한 값에 수렴하게 해주는지에 따라 다르기 때문이라는 점이다. 일반적인 Gradient Descent의 경우 빠르지만 Local Optimum에 갇힐 우려가 있고 이전 step의 관성의 영향을 받지 않고 새롭게 계산된 방향으로만 최적화하는 경향이 있는 Optimizer이고, ADAM의 경우 Momentom과 RMSProp을 합친 방법으로 이전 step에서의 관성을 어느정도 유지하며, 계속해서 계산해온 기울기의 지수 평균을 저장하여 사용하는 방식이므로 Local Optimum에 빠지는 위험은 덜하나 수렴하는 데에 더 많은 step과 시간이 소요될 수 있다.

위 두가지 Optimizer를 비교하여 생각해보자면, 단순 Gradient Descent를 사용할 때는 처음 **W**와 **b**를 초기화 해줄 때 아주 작은 범위 안에서 랜덤한 숫자로 초기화 해주기 때문에 Gradient를 계산하는 과정에서 무작위로 Local Optimum에 갇힐 확률이 크지 않다고 생각하고, 또한 Local Optimum에 갇히더라도 그 수렴한 **W**와 **b**값이 충분히 높은 정확성을 낸다면 더이상 Local Optimum이라 생각하지 않아도 된다고 생각한다. 이 의미는 결국 초기화 범위에 크게 변동이 없는 경우 Gradient Descent Optimizer가 빠르게 정확한 솔루션을 찾을 수 있다는 것이다.

반면에 ADAM의 경우 이전 step에서의 관성을 고려하고 계산해온 기울기의 제곱값의 지수평균을 이용한 계산을 하게 되므로 단순한 Gradient Descent 보다 한 Step에서의 연산량이 많아 더 느려진다 생각하고, 관성을 고려하는 특성 때문에 한 step에 업데이트되는 양과 실제 gradient의 방향 간에 약간의 차이가 존재하므로 정확한 수렴을 위해 더 많은 양의 step이 필요해진다고 생각한다. 이러한 점 때문에 ADAM Optimizer가 GD Optimizer보다 정확성 부분에 있어서 좋지 않은 결과가 나온다고 생각하고, 이러한 점은 ADAM Optimizer의 인자로 들어가는 Beta값과 Epsilon을 조정하면 충분히 더 높은 정확성을 내는 방향으로 개선시킬 수 있다고 생각한다.