ITE4053 Deep Learning course

Practice 2

2016026080 한다진

* Experiment environments

OS Ubuntu 18.04 LTS  
GPU TITAN Xp

Language python3

Tools Jupyter notebook

* Time comparison (element-wise version vs. vectorized version, (m, K) = (1000,100))

스크린샷이(가) 표시된 사진

자동 생성된 설명

Vector 연산을 하는 것이 속도면에서 50배 가량 빠른것을 확인했습니다.

연산시간 외의 정확도 부분에서 거의 동일한 성능을 보였기에 정확하게 테스트했음을 확인했습니다.

* Empirically determined (best) hyper parameter, 𝛼

스크린샷이(가) 표시된 사진

자동 생성된 설명

Runs=100 으로 설정하여 training 을 100번 돌렸고, 그 결과값의 평균을 출력했습니다. Outlier의 성질을 지니고 있는 케이스의 영향을 덜 받기 위해, 여러번 돌렸을떄의 평균값을 사용하여 파라미터의 변화에 따른 모델의 성능변화를 확인하고자 했습니다.

구간별로 더 잘 탐색하기 위해, learning rate 값을 linear하게 나누지 않고, log를 씌운 값이 linear하도록 설정하여 테스트했습니다.

1e-4 근처의 값에서 가장 높은 성능을 보일것으로 예측하였으나, 가장 높은 성능을 내는 Learning rate는 1.0 근처의 값임을 확인했습니다. 1e-4라는 값은 이번 practice#2를 100번의 iteration 만에 학습시키기엔 작은 값이라는 것을 확인했습니다. K 값을 늘린다면, 더 낮은 lr값에 대해서도 정확도가 크게 향상될 것으로 보입니다.

* Accuracy (fill in the blanks in the tables below and add them to the report)

스크린샷이(가) 표시된 사진

자동 생성된 설명

Runs=100 으로 설정하여 training 을 100번 돌렸고, 그 결과값의 평균을 출력했습니다. Outlier의 성질을 지니고 있는 케이스의 영향을 덜 받기 위해, 여러번 돌렸을떄의 평균값을 사용하여 파라미터의 변화에 따른 모델의 성능변화를 확인하고자 했습니다.

전반적으로 M, K값을 늘릴수록 정확도가 향상했습니다. M값이 늘어나면 데이터의 보편적인 특성을 파악하기 쉬워지고 K값이 늘어날수록 데이터를 학습할 수 있는 시간이 늘어나기 때문에 정확도가 느는 것으로 판단했습니다.

n=100

lr=1e-2

|  |  |  |  |
| --- | --- | --- | --- |
| K=100 | m = 10 | m = 100 | m = 1000 |
| train\_loss | 0.14185 | 0.171549 | 0.172270 |
| test\_loss | 0.22862 | 0.174781 | 0.168548 |
| train\_acc | 98.40000 | 98.270000 | 98.918000 |
| test\_acc | 92.31000 | 97.610000 | 98.970000 |

|  |  |  |  |
| --- | --- | --- | --- |
| m=100 | K = 10 | K = 100 | K = 1000 |
| train\_loss | 0.397752 | 0.172736 | 0.074933 |
| test\_loss | 0.405442 | 0.173047 | 0.078105 |
| train\_acc | 96.630000 | 98.280000 | 99.560000 |
| test\_acc | 95.760000 | 97.800000 | 98.960000 |

* Estimated unknown function parameters W & b

스크린샷이(가) 표시된 사진

자동 생성된 설명

가장 높은 성능을 보인 lr, m, K 값을 사용하여 트레이닝을 시키고 그 결과값을 받아왔습니다. w[0]값과 w[1]값이 비슷하게 뭉치는 것을 확인할 수 있었습니다.