과제 #5

M3239.002300_003 데이터 사이언스 특강

Due: 2021년 11월 10일 23시 59분

Numpy를 사용하여 Fashion-MNIST 데이터셋[1]을 학습하는 간단한 MLP 모델을 구현하라. 이번 과제에서는 Michael Nielsen의 딥 러닝 교재[2]에서 제공되는 예제를 참고하여 과제를 구현한다. 문제 1, 2, 3번을 위한 뼈대 코드로 아래와 같은 코드가 주어진다.

- fashion_mnist_loader.py: Fashion MNIST 데이터셋을 읽는 데이터 입출력 코드. 수정하지 말 것.
- run.py: 구현된 네트워크를 사용하여 간단한 테스트를 실행하는 테스트 코드. 수정하지 말 것.
- prob1.py: 문제 1번 구현을 위한 뼈대 코드. TODO 부분만 구현할 것.
- prob2.py: 문제 2번 구현을 위한 뼈대 코드. 원하는 대로 수정/구현할 것.
- prob3.py: 문제 3번 구현을 위한 뼈대 코드. 원하는 대로 수정/구현할 것.

구현 및 실험 결과에 대한 보고서를 작성해야 한다. 구현해야하는 내용과 보고서에 포함되어야 하는 내용은 아래와 같다.

1 문제 1: 기본 MLP 구현

prob1.py의 TODO 부분을 수정하여 class Network의 구현을 완성하라. Network는 activation function으로 sigmoid function을 사용하는 간단한 MLP model이다. 구현 시 출력 예시는 아래와 같다. SGD method의 사용 예시는 run.py를 참고할 것.

```
spdsta@login01:~/HW5$ python3 run.py prob1
Epoch 0: 7394 / 10000
Epoch 1: 7907 / 10000
Epoch 2: 8069 / 10000
Epoch 3: 8140 / 10000
Epoch 4: 8232 / 10000
```

해당 부분을 직접 구현해도 되며, Michael Nielsen 교재의 코드를 사용해도 된다. 보고서에는 아래 항목에 대한 설명이 포함되어야 한다.

- 학습 시 Forward computation 결과와 정답 label을 비교하는 부분은 어디인가? 코드 위치와 내용을 간략히 설명할 것.
- 가장 마지막 layer(=output layer) output의 gradient를 계산하는 부분은 어디인가? 코드 위치와 내용을 간략히 설명할 것.

[1] https://github.com/zalandoresearch/fashion-mnist
[2] http://neuralnetworksanddeeplearning.com/

- 상기한 gradient로부터 마지막에서 두 번째 layer(=마지막 hidden layer) output의 gradient를 계산하는 부분은 어디인가? 코드 위치, 해당 코드에서 각 변수의 의미, 연산의 의미를 설명할 것. 특히 transpose 연산자가 사용되는 경우 왜 transpose를 적용해야 하는지 설명할 것. 필요 시 그림을 포함하여 설명할 것.
- 마지막 layer의 weight의 gradient를 계산하는 부분은 어디인가? 코드 위치, 해당 코드에서 각 변수의 의미, 연산의 의미를 설명할 것.
- 각 layer의 weight를 update하는 부분은 어디인가? 코드 위치, 해당 코드에서 각 변수의 의미, 연산의 의미를 설명할 것.

2 문제 2: Fine-tuning 구현

Fine-tuning 기능을 지원하는 class FTNetwork를 구현하라. FTNetwork는 tuning 인자가 추가된 SGD method를 지원한다. tuning이 True일 경우, 학습 시 **가장 마지막 layer의 weight/bias만 업데이트**한다. 구현 시 출력 예시는 아래와 같다.

```
spdsta@login01:~/HW5$ python3 run.py prob2
==Full training==
Epoch 0: 7394 / 10000
Epoch 1: 7907 / 10000
Epoch 2: 8069 / 10000
==Fine-tuning==
Epoch 0: 8149 / 10000
Epoch 1: 8233 / 10000
Epoch 2: 8232 / 10000
```

prob1.py의 구현을 prob2.py에 복사한 후 수정하여 fine-tuning 기능을 구현하여도 된다. 보고서에는 아래 항목에 대한 설명이 포함되어야 한다.

- tuning 인자가 True로 주어질 경우 동작이 달라지는 부분은 어디인가? 이 경우 왜 마지막 layer의 weight/bias만 업데이트되는가? 간략히 설명할 것.

3 문제 3: Momentum SGD 구현

Momentum을 사용하는 SGD 기능을 지원하는 class MomentumNetwork를 구현하라. Momentum SGD란 매번 weight를 업데이트 할 시 이전 단계에서 weight를 변화시킨 정도와 이번 단계에서 계산한 gradient를 동시에 고려하는 SGD이다. 구체적인 계산 방법은 PyTorch SGD의 Momentum과 동일한 방법을 사용한다. 식은 아래와 같다.

$$v_{t+1} = \mu * v_t + g_{t+1}$$

$$p_{t+1} = p_t + lr * v_{t+1}$$

위 식에서 p_t는 현재의 파라미터(weight, bias), p_{t+1}은 업데이트 후의 파라미터, g_{t+1}은 이번 단계에서 계산된 gradient, lr은 learning rate, μ는 momentum factor를 나타낸다.

구현 시 출력 예시는 아래와 같다.

```
spdsta@login01:~/HW5$ python3 run.py prob3
Epoch 0: 7518 / 10000
Epoch 1: 7932 / 10000
Epoch 2: 8039 / 10000
Epoch 3: 8135 / 10000
Epoch 4: 8181 / 10000
```

prob1.py의 구현을 prob3.py에 복사한 후 수정하여 Momentum SGD 기능을 구현하여도 된다. 보고서에는 아래 항목에 대한 설명이 포함되어야 한다.

- 기존 SGD와 비교해서 달라진 부분은 어디인가? 바뀐 코드와 동작을 간략히 설명할 것.

4 구현 및 제출

실습 서버의 실습 계정 홈 디렉토리의 HW5 디렉토리에 prob1.py, prob2.py, prob3.py, report.pdf를 작성하여 저장한다. 각각의 파일에는 아래와 같은 내용이 구현되어 있어야 한다.

파일	설명
prob1.py	Network class 구현
prob2.py	FTNetwork class 구현
prob3.py	MomentumNetwork class 구현

제출 시 아래와 같은 사항을 유의하라.

- 제출 파일에는 **반드시 class를 정의하는 코드만 포함되어야 하며, 테스트 코드 등이 포함되어서는 안 된다**. 제출한 코드에 불필요한 내용이 포함되어 채점이 방해될 경우 해당 과제가 0점 처리될 수 있다. 단, class에 추가로 method를 구현하여 내부 구현에서 사용하는 것은 괜찮다.
- 실행 결과가 과제 문서의 예시와 완전히 동일할 필요는 없다(즉, accuracy가 예시와 다르게 나타나도 된다). 실행 결과는 구현한 class가 올바르게 동작하는지를 대략적으로 판별하기 위해 사용된다. 구체적으로 문제 스펙에 맞게 코드를 구현했는지는 보고서 내용을 중심으로 판별한다.
- 보고서의 분량을 지나치게 길게 작성할 필요는 없다. 보고서는 본인이 backpropagation의 동작을 제대로 이해하고 있다는 것을 보일 수 있는 수준으로만 작성하면 된다.
- 실습 서버의 HW5 디렉토리에 파일을 저장하면, 과제 제출 시간에 자동으로 해당 파일을 복사해 간다. 파일명을 다른 이름으로 수정할 경우 (예: p1.py) 과제가 제출되지 않을 수 있으니 유의하여야 한다.
- 그레이스 데이를 사용할 경우 spdsta@aces.snu.ac.kr로 이를 알려야 하며, 메일 제목은 [SPDS] HW5-Graceday 학번 이름의 포맷으로 작성하라 (예: [SPDS] HW5-Graceday 2020-20000 안규수).