[인공지능] exercise 보고서

2018920002 컴퓨터과학부 고다현

1. model을 재정의하기 위해 어떤 부분을 수정했는가?

- model 함수

```
def model(t_u, w1, w2, b):
  return w2 * t_u ** 2 + w1 * t_u + b
```

- params

```
params = torch.tensor([1.0, 1.0, 0.0], requires_grad=True)
```

2. 모델 수정과 무관한 부분은?

1번에서 언급한 model 함수와 params 정의, 적절한 learning rate, 적합한 optimizer 외에 모든 것이 무관하다.

3. training 후 loss가 높아지는가 낮아지는가?

전체 data set을 모두 training set으로 사용했을 때

- optimizer를 SGD로 했을 때의 결과

```
params = torch.tensor([1.0, 1.0, 0.0], requires_grad=True)
learning_rate = 1e-2
optimizer = optim.SGD([params], lr=learning_rate) # <1>

training_loop(
    n_epochs = 5000,
    optimizer = optimizer,
    params = params, # <1>
    t_u = t_un,
    t_c = t_c)
```

위의 코드를 실행했을 때 결과는 아래와 같다.

```
Epoch 500, Loss nan
Epoch 1000, Loss nan
Epoch 1500, Loss nan
Epoch 2000, Loss nan
Epoch 2500, Loss nan
Epoch 3000, Loss nan
Epoch 3500, Loss nan
Epoch 4000, Loss nan
Epoch 4500, Loss nan
Epoch 5000, Loss nan
tensor([nan, nan, nan], requires_grad=True)
```

Nan loss가 발생했다. nan loss가 발생한 이유를 다음과 같이 추정했다.

> learning rate 값이 너무 커서 backpropagation 시 weight값이 크게 변해 결국 loss값이 발산한다.

따라서 위 코드에서 learning rate만 1e-2에서 1e-4로 수정한 후 다시 실행해보았고, 결과는 다음과 같다.

```
Epoch 500, Loss 10.708596
Epoch 1000, Loss 8.642083
Epoch 1500, Loss 7.171005
Epoch 2000, Loss 6.123478
Epoch 2500, Loss 5.377227
Epoch 3000, Loss 4.845286
Epoch 3500, Loss 4.465788
Epoch 4000, Loss 4.194724
Epoch 4500, Loss 4.000802
Epoch 5000, Loss 3.861744
tensor([-0.8881,  0.5570, -0.8753], requires_grad=True)
```

epoch가 증가할수록 loss가 점점 감소하는 것을 확인할 수 있다.

- optimizer를 Adam으로 했을 때의 결과

```
params = torch.tensor([1.0, 1.0, 0.0], requires_grad=True)
learning_rate = 1e-1
optimizer = optim.Adam([params], lr=learning_rate) # <1>

training_loop(
    n_epochs = 2000,
    optimizer = optimizer,
    params = params,
    t_u = t_u, # <2>
    t_c = t_c)
```

위의 코드를 실행했을 때 결과는 아래와 같다.

```
Epoch 500, Loss 4.844499
Epoch 1000, Loss 3.785645
Epoch 1500, Loss 3.466678
Epoch 2000, Loss 3.406404
tensor([-0.1081,  0.0060, -1.4510], requires_grad=True)
```

epoch가 증가할수록 loss가 감소했다.

전체 data set을 training set과 validation set으로 나눴을 때

- optimizer를 SGD로 했을 때의 결과

```
params = torch.tensor([1.0, 1.0, 0.0], requires_grad=True)
learning_rate = 1e-2
optimizer = optim.SGD([params], lr=learning_rate)

training_loop(
    n_epochs = 3000,
    optimizer = optimizer,
    params = params,
    train_t_u = train_t_un, # <1>
    val_t_u = val_t_un, # <1>
    train_t_c = train_t_c,
    val_t_c = val_t_c)
```

위의 코드를 실행했을 때 결과는 아래와 같다.

```
Epoch 1, Training loss 715.8550, Validation loss 495.5213
Epoch 2, Training loss 425820.1250, Validation loss 230103.4844
Epoch 3, Training loss 257074800.0000, Validation loss 140169104.0000
Epoch 500, Training loss nan, Validation loss nan
Epoch 1000, Training loss nan, Validation loss nan
Epoch 1500, Training loss nan, Validation loss nan
Epoch 2000, Training loss nan, Validation loss nan
Epoch 2500, Training loss nan, Validation loss nan
Epoch 3000, Training loss nan, Validation loss nan
tensor([nan, nan, nan], requires_grad=True)
```

epoch가 증가할수록 training loss와 validation loss 모두 증가하다가 결국 nan loss가 발생했다. nan loss가 발생한 이유를 다음과 같이 추정했다.

> learning rate 값이 너무 커서 backpropagation 시 weight값이 크게 변해 결국 loss값이 발산한다.

따라서 위 코드에서 learning rate만 1e-2에서 1e-4로 수정한 후 다시 실행해보았고, 결과는 다음과 같다.

```
Epoch 1, Training loss 715.8550, Validation loss 495.5213
Epoch 2, Training loss 401.2646, Validation loss 302.7549
Epoch 3, Training loss 226.9890, Validation loss 191.9145
Epoch 500, Training loss 8.2879, Validation loss 23.7876
Epoch 1000, Training loss 6.6906, Validation loss 20.6075
Epoch 1500, Training loss 5.5689, Validation loss 18.1169
Epoch 2000, Training loss 4.7811, Validation loss 16.1521
Epoch 2500, Training loss 4.2277, Validation loss 14.5912
Epoch 3000, Training loss 3.8389, Validation loss 13.3430
tensor([-0.3624,  0.4718, -0.5125], requires_grad=True)
```

epoch가 증가할수록 training loss와 validation loss 모두 감소하는 것을 확인할 수 있었다. 아직 overfitting이 일어날 기미가 보이지 않으므로 training을 조금 더 수행해봐도 나쁘지 않다고 판단한다.

4. 실제 결과는 더 좋아지는가 나빠지는가?

- 수정 전 모델을 사용했을 때

  optimizer을 SGD로, learning rate를 1e-2로 했을 때의 결과는 다음과 같다.

  ```
  Epoch 1, Training loss 53.5444, Validation loss 201.0541
  Epoch 2, Training loss 30.5667, Validation loss 115.4398
  Epoch 3, Training loss 25.5510, Validation loss 85.4571
  Epoch 500, Training loss 7.7695, Validation loss 21.5329
  Epoch 1000, Training loss 4.1600, Validation loss 9.2849
  Epoch 1500, Training loss 3.3756, Validation loss 5.3324
  Epoch 2000, Training loss 3.2051, Validation loss 3.8717
  Epoch 2500, Training loss 3.1681, Validation loss 3.2738
  Epoch 3000, Training loss 3.1600, Validation loss 3.0131
  tensor([  5.0943, -16.0027], requires_grad=True)
  ```

  optimizer을 SGD로, learning rate를 1e_4로 했을 때의 결과는 다음과 같다.

  ```
  Epoch 1, Training loss 53.5719, Validation loss 200.9305
  Epoch 2, Training loss 53.1893, Validation loss 199.9900
  Epoch 3, Training loss 52.8109, Validation loss 199.0579
  Epoch 500, Training loss 19.2756, Validation loss 86.4042
  Epoch 1000, Training loss 18.9396, Validation loss 81.5487
  Epoch 1500, Training loss 18.7312, Validation loss 80.5227
  Epoch 2000, Training loss 18.5259, Validation loss 79.7249
  Epoch 2500, Training loss 18.3233, Validation loss 78.9493
  Epoch 3000, Training loss 18.1234, Validation loss 78.1829
  tensor([ 2.1980, -0.3975], requires_grad=True)
  ```

  learning rate가 1e_2일 때와 비교했을 때 training loss와 validation loss 모두 더 크다.

- 수정 후 모델을 사용했을 때

  optimizer을 SGD로, learning rate를 1e-2로 했을 때의 결과는 다음과 같다.

  ```
  Epoch 1, Training loss 715.8550, Validation loss 495.5213
  Epoch 2, Training loss 425820.1250, Validation loss 230103.4844
  Epoch 3, Training loss 257074800.0000, Validation loss 140169104.0000
  Epoch 500, Training loss nan, Validation loss nan
  Epoch 1000, Training loss nan, Validation loss nan
  Epoch 1500, Training loss nan, Validation loss nan
  Epoch 2000, Training loss nan, Validation loss nan
  Epoch 2500, Training loss nan, Validation loss nan
  Epoch 3000, Training loss nan, Validation loss nan
  tensor([nan, nan, nan], requires_grad=True)
  ```

  수정 전 모델과 비교했을 때 training loss와 validation loss 모두 크다. 따라서 수정 후의 실제 결과가 더 나쁘다고 할 수 있다.

optimizer을 SGD로, learning rate를 1e-4로 했을 때의 결과는 다음과 같다.

```
Epoch 1, Training loss 715.8550, Validation loss 495.5213
Epoch 2, Training loss 401.2646, Validation loss 302.7549
Epoch 3, Training loss 226.9890, Validation loss 191.9145
Epoch 500, Training loss 8.2879, Validation loss 23.7876
Epoch 1000, Training loss 6.6906, Validation loss 20.6075
Epoch 1500, Training loss 5.5689, Validation loss 18.1169
Epoch 2000, Training loss 4.7811, Validation loss 16.1521
Epoch 2500, Training loss 4.2277, Validation loss 14.5912
Epoch 3000, Training loss 3.8389, Validation loss 13.3430
tensor([-0.3624,  0.4718, -0.5125], requires_grad=True)
```

수정 전 모델과 비교했을 때 training loss와 validation loss 모두 작다. 또 learning rate가 1e-2일 때의 수정 후 모델과 비교했을 때도 두 loss 모두 작다.

따라서 다음과 같은 결론을 도출할 수 있다.

> learning rate가 1e-2일 때는 수정 후 모델의 실제 결과는 수정 전보다 더 나쁘다. 수정 후 모델의 성능을 좋게 유지하려면 learning rate를 줄여야 한다.