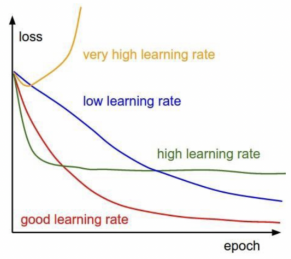
MNIST 성능 올리기

epoch, learning rate, layer, batch size, optimizer 바꾸었을 때 accuracy, loss 에 대한 분석

🧐 어떻게 최적의 네트워크를 찾는가?

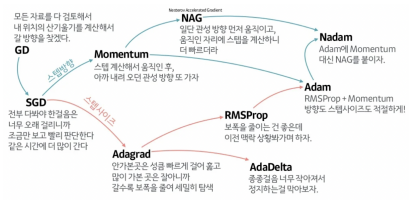
- 작은 크기의 모형부터 시작한다. (overfitting의 가능성이 적다.) 네트워크의 크기 (layer의 수 증가, ~~layer의 unit수의 증가~~) 를 증가시키면서 검증과정을 거친다.
- Learning rate 을 조절하면서 성능을 향상시킨다. learning rate가 너무크면 수렴을 못할 가능성이 있고 너무 작으면 local minimun에 빠져 못나올 수 있음



- epoch : epoch 을 높일 수록 다양한 무작위 가중치로 학습을 해보기 때문에 적합한 파라미터를 찾을 확률이 올라가게되어 손실이 적어지게된다. 하지만 지나치게 높은 epoch은 그 학습 데이터셋에 오버피팅되어 다른 데이터셋에 대한 제대로된 예측이 어렵다.
- batch size : 연산 한번에 들어가는 데이터의 크기. 배치 사이즈가 너무 크면 한번에 처리할 데이터양이 많아져 학습속도가 느리고 메모리 부족 문제 발생. 배치사이즈가 너무 적으면 적은 데이터를 대상으로 가중치를 업데이트 하고 업데이트가 자주 발생하게되어 훈련이 불안정해짐
- optimizer :

SGD 를 사용하니 Train Loss 값이 이리저리 튀는 부분이 있다. 왜 그럴까?

→ SGD의 경우, 전체 데이터가 아닌 일부, 즉 Mini Batch를 사용하기 때문에, 값을 랜덤하게 주게 되어 loss 값이 조금씩 튀게 된다.



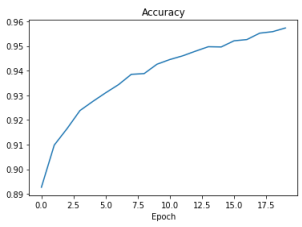
💡 Trial 1 (기본)

epoch : 20 , batch size : 16 , learning rate : 0.0001, optimizer : Adam , layer :

```
nn.Linear(28 * 28, 64),
nn.ReLU(),
nn.Linear(64, 10),
nn.Sigmoid()
```

결과

Accuracy : 0.9573 | Training Loss : 1.46425



→ 수정할 parameter

1. noise 감소를 위해 batch size를 기존 값 16에서 32으로 증가.
2. layer , optimizer, learning rate, epoch 그대로

💡 Trial 2 (batchsize 늘리기)

epoch : 20 , batch size : 32 , learning rate : 0.0001, optimizer : Adam

결과

Accuracy : 0.9473 | Training Loss : 1.4803

- batch size만 늘려봤는데 미미하긴하지만 accuracy와 training loss 결과가 안좋아짐 ㅜㅜ. Underfitting인 것 같아서 epoch에 변화를 주어봄.

→ 수정할 parameter

1. noise 감소를 위해 batch size를 기존 값 16에서 32으로 증가.
2. **underfitting 가능성이 있다고 판단되어 epoch을 50으로 증가**.
3. layer , optimizer , learning rate 그대로

💡 Trial 3 (epoch 늘리기)

epoch : 50 , batch size : 32 , learning rate : 0.0001, optimizer : Adam

결과

Accuracy : 0.9679 | Training Loss : 1.4684

- epoch을 늘려 학습을 더 시키니 accuracy는 올라가고 training loss는 떨어짐 😆

→ 수정할 parameter

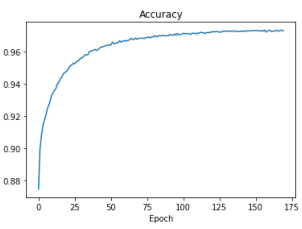
1. **epoch을 100으로 증가**.
2. layer , optimizer , learning rate 그대로

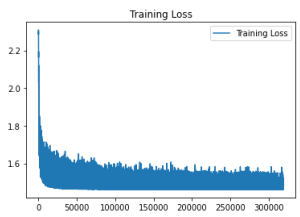
💡 Trial 4 (epoch 더ㅓ 늘리기)

epoch : 100 , batch size : 32 , learning rate : 0.0001, optimizer : Adam

결과

Accuracy : 0.9728 | Training Loss : 1.4671





- epoch만 늘려서 성능을 올렸다. 그런데 드라마틱하진 않고 단순히 epoch 을 키우는 것 만으로는 더 이상의 성능 향상이 기대되지 않는다. 다른 파라미터의 조정이 필요할 것같다. 하지만 성능향상이 목표가 아니라 파라미터 조정에 따른 성능비교가 중점이니 다음 시도는 epoch을 대폭줄여보고 다른 파라미터의 조정을 해보겠다.

→ 수정할 parameter

1. **epoch을 15로 감소**.
2. layer 조정 (차원 조절, ReLU함수 사용) , batch 늘리기, learning rate 0.001로 증가.

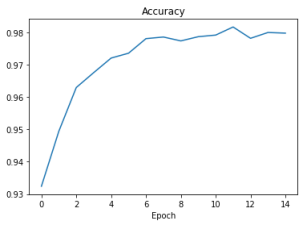
💡 Trial 5 (epoch을 많이 줄이고 batch와 layer, learning rate 늘려보기)

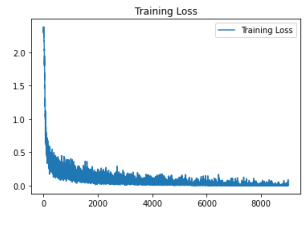
epoch : 15 , batch size : 100 , learning rate : 0.001, optimizer : Adam ,layer :

```
nn.Linear(28 * 28, 512),
nn.ReLU(),
nn.Linear(512, 256),
nn.ReLU(),
nn.Linear(256,10)
```

결과

Accuracy : 0.9797 | Training Loss : 0.026 (???)





- training loss가 대폭 줄었다.. Adam과 batch size의 관계가 있나 싶어서 구글링 하다 찾은건데

“ 하지만 요즘엔 Adam optimizer나 batch normalization(BN)등의 기법을 사용함으로써 학습안정화가 정말 잘 되는 네트워크가 많습니다. 따라서 batch size가 커도 local minimum을 잘 지나쳐 global minimum으로 수렴할 수 있게되었습니다. 따라서 **최근에는 batch size를 줄 수 있는만큼 최대한 크게줘야** 좋다고 할 수 있습니다. “ (garam.log)

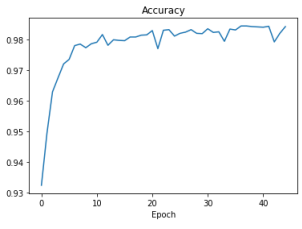
라고한다. Adam optimizer의 사용으로 배치를 100정도 크게하여도 loss가 아주아주 줄어든게 아닌가 싶다.

→ 수정할 parameter

1. **epoch을 30으로 증가**.
2. layer , optimizer , learning rate 그대로

💡 Trial 6 (trial 5에서 epoch 늘리기)

epoch : 30 , batch size : 100 , learning rate : 0.001, optimizer : Adam

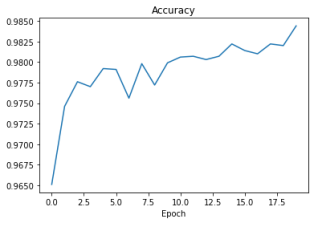


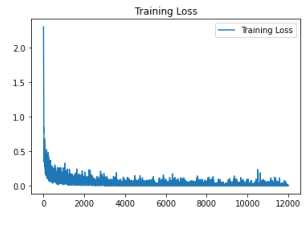
- accuracy도 loss도 불안정해졌다. ㅠㅠ 기존의 파라미터에서 epoch을 20으로도 해봐야겠다.

💡 Trial 7 (최종, 적절한 epoch조절)

epoch : 20 , batch size : 100 , learning rate : 0.001, optimizer : Adam

accuracy: 0.9844 | loss : 0.0024





- epoch을 적절히 조정하여 가장 좋은 성능을 만들어냈다.

epoch , batch, learning rate, layer 조절만으로 성능을 0.95에서 0.98까지 끌어올릴 수 있다. 많은 조정을 시도해보아야 좋은 결과를 얻을 수 있는 것 같다…