소프트웨어융합학과
2020111406 이해현

〔과제 내용〕

Mnist 데이터를 사용해서 CNN 구조를 구현하고, 과적합을 방지하기 위해 dropout을 적용하여 결과 분석

- 모델 구성
    - Convolution layer 2개 (각 레이어 마다 3x3 kernel, 2x2 max-pooling 적용)
    - Fully-connected layer 2개 (각각 256개의 노드 수로 구성, dropout(0.8) 적용)

- loss function : Cross Entropy
- Optimizer : ①AdamOptimizer ②SGD 비교하기

〔실험 내용 및 결과 분석〕

위의 실험 조건에 batch size는 256, learning rate는 0.02, epoch수는 10으로 주요 학습 파라미터를 설정하여 실험을 진행함.

1. Optimizer SGD 사용 실험 결과

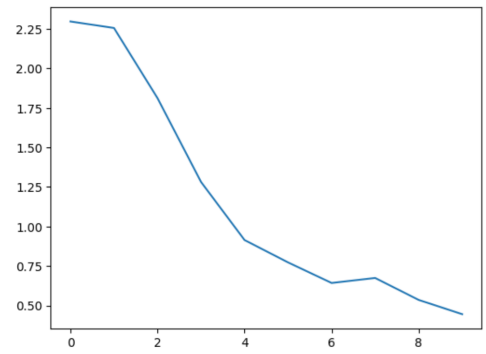

그림 1 SGD - loss function 시각화 결과

```
tensor(2.2973, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(2.2563, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(1.8124, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(1.2818, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.9142, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.7726, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.6426, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.6742, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.5353, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.4451, device='cuda:0', grad_fn=<NllLossBackward0>)
```

그림 2 SGD - 학습 중 loss 값 기록

Optimizer로 SGD를 사용해 CNN 모델을 학습하여 loss function을 시각화한 결과, 약 0.5 이하로 loss 값이 일정하게 감소함을 확인 할 수 있었고, Accuracy를 측정하기 위해 Test data로 prediction을 진행한 결과, 87.02424621582031%의 정확도를 보였다.

2. Optimizer Adam 사용 실험 결과

2-1) Optimizer Adam 실험 1

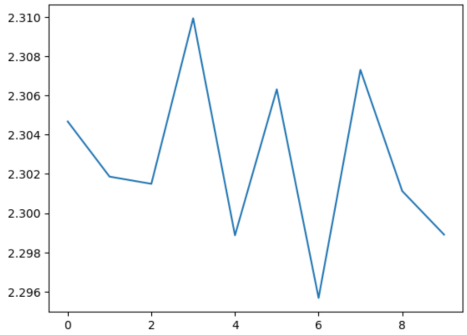

그림 3 Adam 실험1 - loss function 시각화

Optimizer SGD와 같은 학습 파라미터 조건으로 Optimizer로 Adam를 사용해 CNN 모델을 학습하여 loss function을 시각화한 결과, 약 2.2정도로밖에 loss 값이 감소하지 못했고, 감소하는 과정에서도 일정하게 감소하지 못하고, Spike가 있었음을 확인할 수 있었다. Accuracy를 측정하기 위해 Test data로 prediction을 진행한 결과, 9.755609512329102%로 매우 낮게 정확도가 측정되었다.

2-2) Optimizer Adam 실험 2

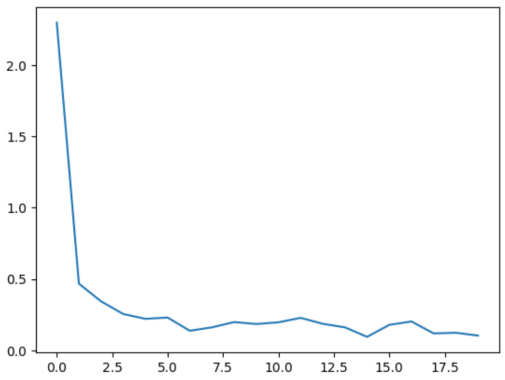

그림 4 Adam 실험2 - loss function 시각화

```
tensor(2.2985, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.4665, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.3423, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.2536, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.2197, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.2288, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.1361, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.1598, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.1971, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.1835, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.1953, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.2264, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.1851, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.1599, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.0933, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.1776, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.2014, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.1174, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.1219, device='cuda:0', grad_fn=<NllLossBackward0>)
tensor(0.1023, device='cuda:0', grad_fn=<NllLossBackward0>)
```

그림 5 Adam 실험2 - 학습 중 loss 값 기록

따라서 learning rate를 0.002, epoch 수를 20으로 파라미터 값을 수정하여 다시 학습을 하여 loss function을 시각화한 결과, 0.1023으로 loss 값이 실험 2-1에 비해 비교적 일정하게 감소함을 확인 할 수 있었다. 또한, Test data로 prediction을 진행하였을 때도 95.45940399169922% 의 정확도를 기록하여 실험 2-1에 비해 높은 성능을 보여주었다.

3. 실험 결과 분석

learning rate 0.02, epoch 10 로 동일한 실험 조건에서 SGD를 optimizer로 사용한 〔실험 1〕과, Adam을 optimizer로 사용한 〔실험 2-1〕의 결과를 비교했을 때, Adam을 사용했을 때, 훨씬 정확도 결과가 낮게 측정되었다. 이는 각 가중치에 대해 동일한 학습률 사용하는 SGD와 달리, 각 가중치에 대해 개별적으로 학습하여 비교적 빠르게 수렴하는 Adam Optimizer 특성을 고려하지 못하고 learning rate값을 크게 설정하고, epoch 값을 작게 설정해서 너무 빨리 수렴을 하게 되어 과적합이 발생한 것으로 보인다.

따라서, learning rate 값을 0.002, epoch 20으로 파라미터 값을 조정하여 〔실험 2-2〕를 진행한 결과, loss 값이 2.0에서 약 0.1로 큰 폭으로 일정하게 감소하였고, 정확도 또한 약 95%이상으로 측정되어 SGD를 사용한 〔실험 1〕에 비해 높은 성능 향상을 보였다.

따라서, SGD는 Adam에 비해서 수렴속도가 느리지만, 훈련 데이터가 적거나, 모델 구조가 간단한 경우 훈련 데이터에 과적합되는 경향이 적어 안정적인 일반화를 제공할 수 있다. Adam은 SGD 보다 훨씬 빠르게 수렴을 제공하지만, 때때로 간단한 모델에서 과적합을 발생시킬 위험이 있으므로 파라미터 조정을 통해 과적합을 방지하여야 함을 알 수 있었다.