

5장. 오차역전파법(3) 실습문제

신경망

학번: 202404178 이름: 강우현

1. 5.7.2 절(슬라이드 13)에서 TwoLayerNet 클래스에서 굵은 글씨로 표시된 부분이 세 군데 있다. 코드를 모두 옮겨 적고 각 줄마다 주석으로 각 줄의 설명(의미, 기능 등)을 추가하시오.

(1) `__init__` 함수 부분:

```
# 계층 저장용 OrderedDict 생성
self.layers = OrderedDict()

# 첫 번째 Affine 계층 생성
self.layers['Affine1'] = Affine(self.params['W1'], self.params['b1'])

# 활성화 함수 계층
self.layers['Relu1'] = Relu()

# 두 번째 Affine 계층
self.layers['Affine2'] = Affine(self.params['W2'], self.params['b2'])

# 소프트맥스 + 손실 함수(교차 엔트로피)
self.lastLayer = SoftmaxWithLoss()
```

(2) `predict` 함수 부분:

```
# 입력 데이터를 받아 각 계층을 순서대로 통과시키며 출력 계산
for layer in self.layers.values():
    x = layer.forward(x) # forward 함수를 통해 계층 순서대로 진행
```

(3) `gradient` 함수 부분:

```
# 미분값은 항상 1 고정
dout = 1

# SoftmaxWithLoss 계층의 역전파 수행
dout = self.lastLayer.backward(dout)

# 역전파 순서대로 처리
layers = list(self.layers.values())
layers.reverse()

# 미분값을 역전파로
for layer in layers:
    dout = layer.backward(dout)
```

2. 5.7.3 절의 결론(알려 주고자 하는 내용)을 간단히 설명하시오.

오차역전파법은 빠르고 효율적인 기울기 계산이 가능하지만 구현이 까다로워 오류가 생기기 쉽다. 이에 따라 수치 미분은 학습용이 아니라, 오차역전파법의 정확성을 확인하는 데 사용된다.

3. 5.7.4 절 코드에 대해 다음 물음에 답하시오.

(1) 이 코드의 동작 순서(절차, 흐름)를 글(비교적 편한 언어)로 설명하시오.

먼저 데이터를 불러와 정규화한 뒤, 입력층 784개·은닉층 50개·출력층 10개로 구성된 신경망을 초기화한다. 이후 정해진 횟수만큼 반복하면서 매번 학습 데이터에서 무작위로 미니배치를 선택하고, 역전파를 이용해 가중치의 기울기를 계산한다. 계산된 기울기에 학습률을 적용하여 가중치와 편향을 갱신하고, 손실값을 기록하며 일정 주기마다 학습 데이터와 테스트 데이터의 정확도를 출력한다. 이를 통해 신경망의 숫자 인식 성능이 향상된다.

(2) loss와 accuracy를 누적하여 저장하였다. 저장한 이유는?

학습이 원하는대로 잘 진행되고 있는지 확인하기 위해 누적하여 계산한다.

(3) 이 코드를 타이핑하고 실행하시오. 코드와 출력 결과를 캡처하여 붙여넣기 하시오.

```
train acc: 0.13798333333333335
test acc: 0.1393
train acc: 0.8995833333333333
test acc: 0.9025
train acc: 0.9190333333333334
test acc: 0.9213
train acc: 0.9332833333333334
test acc: 0.932
train acc: 0.9400333333333334
test acc: 0.9378
train acc: 0.9483666666666667
test acc: 0.9441
train acc: 0.9546166666666667
test acc: 0.9522
train acc: 0.9593833333333334
test acc: 0.9564
train acc: 0.963833333333333
test acc: 0.9602
train acc: 0.966983333333333
test acc: 0.9619
train acc: 0.9698166666666667
test acc: 0.9618
train acc: 0.9725333333333334
test acc: 0.9648
train acc: 0.9735666666666667
test acc: 0.9657
train acc: 0.974883333333333
test acc: 0.9653
train acc: 0.9763666666666667
test acc: 0.9673
train acc: 0.9778666666666667
test acc: 0.9696
train acc: 0.9790166666666666
test acc: 0.9696
```