연구지도 및 연구윤리 11주차 보고서

(Chapter5. 오차역전파-2)

경북대학교 전자공학부
2016113566 김남영

1) five layer net 구현 및 결과 비교 (한 줄씩 설명)

```python
class FiveLayerNet:

    def __init__(self, input_size, hidden1_size, hidden2_size, hidden3_size, hidden4_size, output_size, weight_init_std = 0.01):
        # 가중치 초기화
        self.params = {}
        self.params['W1'] = weight_init_std * np.random.randn(input_size, hidden1_size)
        self.params['b1'] = np.zeros(hidden1_size)
        self.params['W2'] = weight_init_std * np.random.randn(hidden1_size, hidden2_size)
        self.params['b2'] = np.zeros(hidden2_size)
        self.params['W3'] = weight_init_std * np.random.randn(hidden2_size, hidden3_size)
        self.params['b3'] = np.zeros(hidden3_size)
        self.params['W4'] = weight_init_std * np.random.randn(hidden3_size, hidden4_size)
        self.params['b4'] = np.zeros(hidden4_size)
        self.params['W5'] = weight_init_std * np.random.randn(hidden4_size, output_size)
        self.params['b5'] = np.zeros(output_size)

        # 계층 생성
        self.layers = OrderedDict()
        self.layers['Affine1'] = Affine(self.params['W1'], self.params['b1'])
        self.layers['Relu1'] = Relu()
        self.layers['Affine2'] = Affine(self.params['W2'], self.params['b2'])
        self.layers['Relu2'] = Relu()
        self.layers['Affine3'] = Affine(self.params['W3'], self.params['b3'])
        self.layers['Relu3'] = Relu()
        self.layers['Affine4'] = Affine(self.params['W4'], self.params['b4'])
        self.layers['Relu4'] = Relu()
        self.layers['Affine5'] = Affine(self.params['W5'], self.params['b5'])

        self.lastLayer = SoftmaxWithLoss() # 교차엔트로피 오차를 포함한 layer

    def predict(self, x):
        for layer in self.layers.values():
            x = layer.forward(x)

        return x

    # x : 입력 데이터, t : 정답 레이블
    def loss(self, x, t):
        y = self.predict(x)
        return self.lastLayer.forward(y, t)

    def accuracy(self, x, t):
        y = self.predict(x)
        y = np.argmax(y, axis=1)
        if t.ndim != 1 : t = np.argmax(t, axis=1)

        accuracy = np.sum(y == t) / float(x.shape[0])
        return accuracy
```

```python
    def gradient(self, x, t):
        # forward
        self.loss(x, t) # 입력 데이터와 정답 레이블을 통해 손실 함수를 구함

        # backward
        dout = 1
        dout = self.lastLayer.backward(dout) # lastLayer는 따로 역전파(layer 이름이 다름)

        layers = list(self.layers.values())
        layers.reverse() # layer 층을 역순으로 바꿈
        for layer in layers: # 역순이 되었으므로 backward 할 수 있음
            dout = layer.backward(dout)

        # 결과 저장
        grads = {}
        grads['W1'], grads['b1'] = self.layers['Affine1'].dW, self.layers['Affine1'].db
        grads['W2'], grads['b2'] = self.layers['Affine2'].dW, self.layers['Affine2'].db
        grads['W3'], grads['b3'] = self.layers['Affine3'].dW, self.layers['Affine3'].db
        grads['W4'], grads['b4'] = self.layers['Affine4'].dW, self.layers['Affine4'].db
        grads['W5'], grads['b5'] = self.layers['Affine5'].dW, self.layers['Affine5'].db

        return grads
```

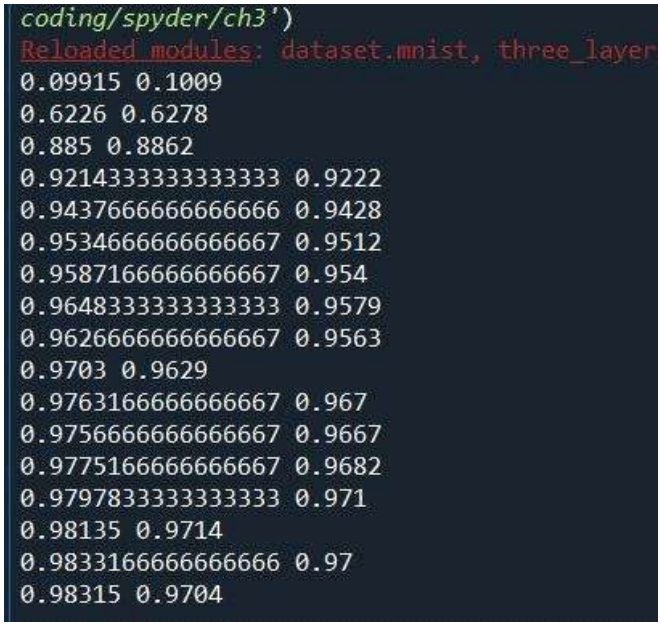
위와 같이 5층 신경망을 구현하였음.

```python
import numpy as np
from dataset.mnist import load_mnist
from five_layer_net import FiveLayerNet

# 데이터 읽기
(x_train, t_train), (x_test, t_test) = load_mnist(normalize=True, one_hot_label=True)

network = FiveLayerNet(input_size=784, hidden1_size=50, hidden2_size=50, hidden3_size=50, hidden4_size=50, output_size=10)
# 레이어 수에 맞게 hidden4 까지 값을 설정해줌

iters_num = 10000 # 갱신 반복 횟수
train_size = x_train.shape[0]
batch_size = 100 # 미니 배치 크기
learning_rate = 0.1 # 학습률

train_loss_list = [] # 손실 함수 값의 변화 추이 비교 목적
train_acc_list = [] # 정확도 평가를 위함
test_acc_list = [] # training 정확도와 비교 목적

iter_per_epoch = max(train_size / batch_size, 1)

for i in range(iters_num):
    batch_mask = np.random.choice(train_size, batch_size) # 설정한 사이즈만큼 랜덤한 dataset 설정
    x_batch = x_train[batch_mask]
    t_batch = t_train[batch_mask]

    # 기울기 계산
    #grad = network.numerical_gradient(x_batch, t_batch) # 수치 미분 방식
    grad = network.gradient(x_batch, t_batch) # 오차역전파법 방식

    # 갱신
    for key in ('W1', 'b1', 'W2', 'b2', 'W3', 'b3', 'W4', 'b4', 'W5', 'b5'):
        network.params[key] -= learning_rate * grad[key]

    loss = network.loss(x_batch, t_batch) # 손실 함수 값을 구함
    train_loss_list.append(loss) #손실 함수 값을 리스트로 저장

    if i % iter_per_epoch == 0:# 에폭당 정확도를 계산해서 값이 어떻게 변화하는지 확인
        train_acc = network.accuracy(x_train, t_train)
        test_acc = network.accuracy(x_test, t_test)
        train_acc_list.append(train_acc)
        test_acc_list.append(test_acc)
        print(train_acc, test_acc)
```

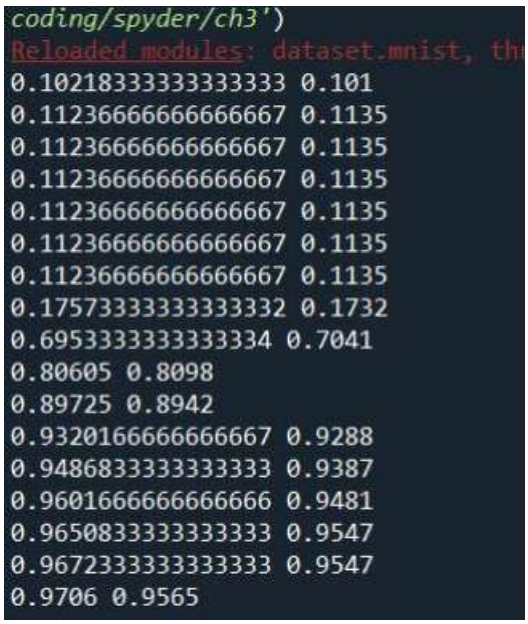
구현한 5층 신경망을 통해 위와 같이 학습을 실행함.

```
coding/spyder/ch3')
Reloaded modules: dataset.mnist, five_layer_net
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.10441666666666667 0.1028
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
0.11236666666666667 0.1135
```

하지만 위와 같이 정확도가 0.1135에서 고정되는 모습을 볼 수 있었다. 원인을 찾기 위해 3층 신경망부터 파라미터만 겹겹이 쌓아가며 결과를 비교해보았다.



3층 신경망 결과 - 잘 실행됨을 알 수 있다.



4층 신경망부터 갱신이 제대로 안되는 모습을 볼 수 있다. 5층 신경망에서 loss값을 print 해보았더니 손실 함수 값이 떨어지지 않고 대략 2.0~2.4에서 머무는 모습을 볼 수 있었다. 똑같은 신경망에서 파라미터만 하나씩 추가하여 레이어를 쌓았지만 어떤 이유에선지 가중치와 편향의 갱신이 올바른 방향으로 이루어지지 않았다.

2) 수치미분 방식 및 오차역전파법 방식 결과 비교

위에서 볼 수 있듯 갱신은 제대로 이루어지지 않았지만, 계산 속도에서는 큰 차이를 보였다. 수치미분으로 기울기를 구했을 때는 프로그램이 실행이 제대로 되는건지도 모를 만큼 느리게 진행되었고 오차역전파법으로 기울기를 구했을 때에는 대략 3초에 한번씩 정확도를 갱신하여 프린트하였다.

3)고찰

1. 같은 방식으로 단지 가중치와 편향 파라미터만 추가하였을 뿐인데, 어떤 이유에선지 4층 신경망, 5층 신경망부터는 제대로 된 갱신이 이루어지지 않았다. 학습률을 고쳐도 보고, hidden layer의 값을 바꾸어도 보았지만 항상 0.1135라는 값으로 고정되어 있었다. 손실 함수가 2.0~2.4에서 머무는 것으로 보아 올바르게 gradient 값이 계산되지 않은 채 가중치와 편향의 갱신이 이루어지는 것 같다. 하지만 2~4층 신경망 코드에서 아무 변화 없이 단지 가중치와 편향 파라미터만 추가 했을 뿐인데 이러한 오류가 발생하는 이유를 모르겠다.

2. 내용이 갈수록 어려워지는 것 같다. 코드를 쓰면서도 이해가 되지 않는 부분이 이전보다 훨씬 많았다. 책을 조금 더 꼼꼼히 보고 이해를 했다면 분명히 오류가 발생한 이유를 찾을 수 있었을텐데, 그런 점이 부족했던 것 같다. 반복해서 책을 보아야겠다.