실습문제: 6장. 학습 관련 기술들(2)

신경망

학번: 202404178 이름: 강우현

제출물: 이 hwp 파일에 답을 적어 제출하세요. 파이썬 코드 제출할 필요 없음.
팀과 협의해서 문제를 푸세요. 이번 실습문제는 팀플에 큰 도움이 될 것입니다.

1. [가중치 초깃값] 가중치의 초깃값을 주는 기본적인 아이디어는, 가중치마다 서로 다른 작은 숫자들을 할당하는 데, 얼마나, 어떻게 작게 줄 것인 지가 문제이다. 교재와 common/multi_layer_net.py 코드를 참조하여 다음에 적절히 답하시오.
   참고 사이트: https://github.com/WegraLee/deep-learning-from-scratch

(1) 신경망 모델의 계층들(Layer)을 생성하는 코드를 찾아 옮겨 붙이시오. 힌트: __init__ 함수 참조

```
self.params = {}

all_size_list = [input_size] + hidden_size_list + [output_size]
for idx in range(1, len(all_size_list)):
    fan_in = all_size_list[idx-1]
    fan_out = all_size_list[idx]

    self.params['W' + str(idx)] = weight_init_std * np.random.randn(fan_in, fan_out)
    self.params['b' + str(idx)] = np.zeros(fan_out)

self.layers = OrderedDict()
for idx in range(1, len(all_size_list)-1):
    self.layers['Affine' + str(idx)] = Affine(self.params['W' + str(idx)], self.params['b' + str(idx)])
    self.layers['Relu' + str(idx)]   = Relu()

idx = len(all_size_list) - 1
self.layers['Affine' + str(idx)] = Affine(self.params['W' + str(idx)], self.params['b' + str(idx)])

self.last_layer = SoftmaxWithLoss()
```

(2) 다음 빈 칸에 적절히 답하시오.

가중치 초깃값 설정 방법	파이썬 코드 (__init_weight 메서드 참조)	권장 사용법(교재 참조)
std=0.01	W = 0.01 * np.random.randn(n_out, n_in)	전통적인 초기화 방법
Xavier	W = np.random.randn(n_out, n_in) * np.sqrt(1.0 / n_in)	sigmoid, tanh에 적합
He	W = np.random.randn(n_out, n_in) * np.sqrt(2.0 / n_in)	ReLU 계열에 최적

(2) 코드에서 편향 b는 어떻게 초기화 했는가?

모든 값을 0으로 초기화하였다.

(3) [그림 6-14]를 보고 세 방법 중, He 초깃값을 사용하는 것이 가장 좋다고 말할 수 있는 이유는?

ReLU 사용 시 He 초기값이 학습 수렴 속도와 최종 성능 모두에서 우수한 결과를 보인다.

2. ch06/weight_init_compare.py 파일에 대해 물음에 답하시오.
(1) 실험을 위해 사용한 신경망 구조를 설명하시오.

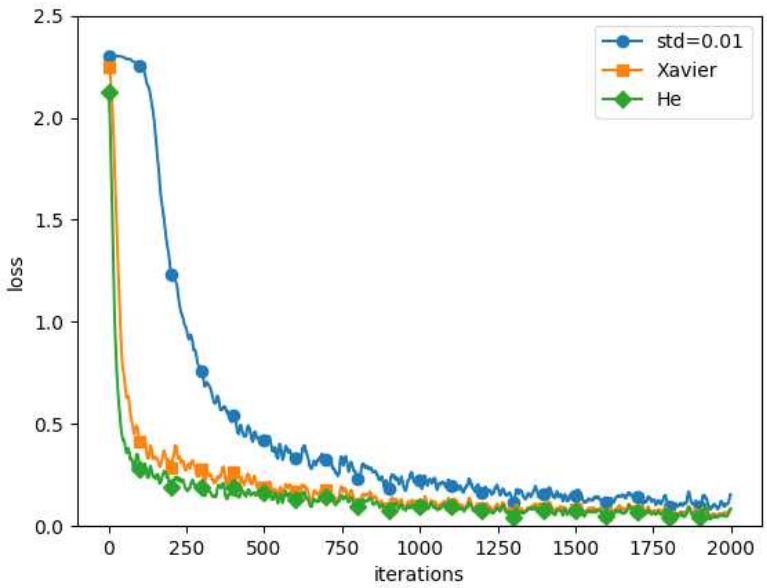
입력층: 784 (MNIST 28×28)
은닉층: 5개 층, 각 층 100개 뉴런
출력층: 10
활성화 함수: ReLU
배치 사이즈: 100, max_epochs=20

(2) optimizer로 Adam을 사용하도록 코드를 수정하고, Xavier, He 초기화시 손실함수 값이 0에 수렴하도록 학습률을 적절히 조정하시오. 코드와 그래프 결과를 옮겨 붙이시오.

```
import sys, os
sys.path.append(os.path.join(os.path.dirname(__file__), '..'))

import numpy as np
import matplotlib.pyplot as plt
from dataset.mnist import load_mnist
from common.util import smooth_curve
from common.multi_layer_net import MultiLayerNet
from common.optimizer import Adam
```

```
(x_train, t_train), (x_test, t_test) = load_mnist(normalize=True)

train_size = x_train.shape[0]
batch_size = 128
max_iterations = 2000

weight_init_types = {'std=0.01': 0.01, 'Xavier': 'sigmoid', 'He': 'relu'}

optimizer = Adam(lr=0.001)

networks = {}
train_loss = {}
for key, weight_type in weight_init_types.items():
    networks[key] = MultiLayerNet(
        input_size=784,
        hidden_size_list=[100, 100, 100, 100],
        output_size=10,
        weight_init_std=weight_type
    )
    train_loss[key] = []



for i in range(max_iterations):
    batch_mask = np.random.choice(train_size, batch_size)
    x_batch = x_train[batch_mask]
    t_batch = t_train[batch_mask]

    for key in weight_init_types.keys():
        grads = networks[key].gradient(x_batch, t_batch)
        optimizer.update(networks[key].params, grads)

        loss = networks[key].loss(x_batch, t_batch)
        train_loss[key].append(loss)

    if i % 100 == 0:
        print("========== iteration:" + str(i) + " ==========")
        for key in weight_init_types.keys():
            loss = networks[key].loss(x_batch, t_batch)
```

```
            print(key + ": " + str(loss))


markers = {'std=0.01': 'o', 'Xavier': 's', 'He': 'D'}
x = np.arange(max_iterations)
for key in weight_init_types.keys():
    plt.plot(x, smooth_curve(train_loss[key]),
             marker=markers[key], markevery=100, label=key)
plt.xlabel("iterations")
plt.ylabel("loss")
plt.ylim(0, 2.5)
plt.legend()
plt.show()
```



3. [배치 정규화] 배치 정규화를 사용하면, [그림 6-19]에서 보듯이, 학습이 빨리 되어 학습시간(파라미터가 수렴하는데 필요한 에폭 수)을 줄일 수 있다(적은 에폭만 학습해도 꽤 괜찮은 파라미터를 구할 수 있다)고 알려져 있다.

(1) 배치 정규화의 기본 아이디어는 무엇인가?

내부 공변량 변화를 줄여 학습을 빠르게, 초깃값에 덜 민감하게 만들기 위해

(2) 배치 정규화는 언제(무엇과 무엇 사이에) 하는가?

일반적으로 Affine(선형 변환) 계층과 활성화 함수 사이에 삽입한다.

(3) 배치 정규화는 두 가지 스탭(step)으로 구성되어 있다. 무엇인가? 필요한 수식을 적으시오.

정규화와 스케일 및 쉬프트 단계이다.

정규화 :

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$$
$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$$
$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

스케일 및 쉬프트 :

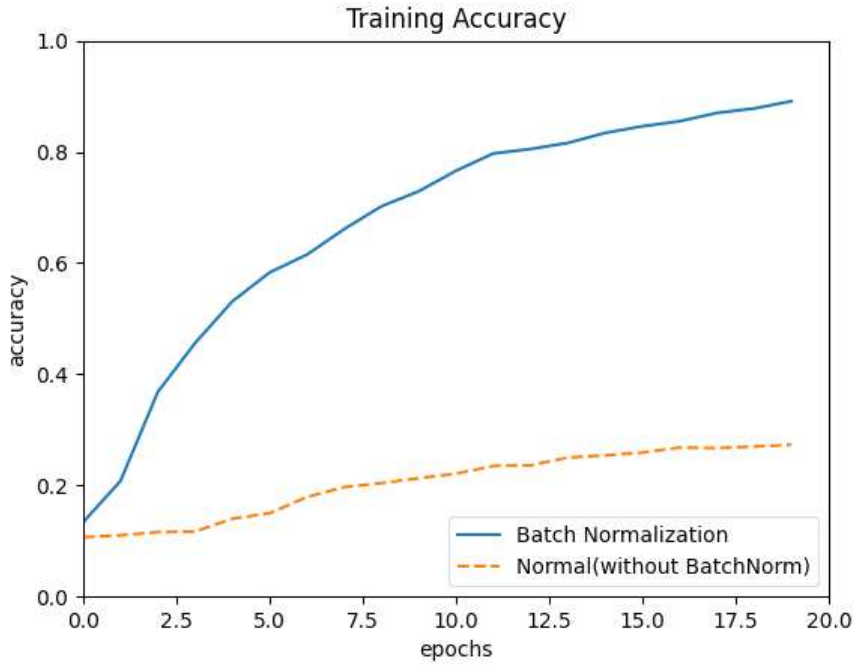
$$y_i = \gamma \hat{x}_i + \beta$$

(4) 배치 정규화를 위해 데이터를 사용하여 학습하는 매개변수가 두 개있다. 무엇인가?

γ (gamma) : 스케일 파라미터

β (beta) : 시프트(편향) 파라미터

(5) batch_norm_test.py 코드를 실행하고 출력된 그래프를 옮겨 붙이시오.



(6) 출력 그래프를 보고 배치 정규화를 사용할 때 학습이 빠르게 진행된다고 말할 수 있는 이유를 간단히 적으시오.

파라미터가 수렴하는 데 필요한 에폭 수가 줄어들어서