인공지능개론 7장(하)-8장 정리노트#5

(꼭 PDF로 변환후 제출하기)

ICT 융합공학부 20학번 박준형 / ICT 융합공학부 20학번 김정호

7장(하)

준형: 이번 장에서는 합성곱 신경망(CNN)에 대해 많이 배웠잖아. 합성곱 계층과 풀링 계층을 직접 구현해보면서 4차원 배열도 다뤘고, im2col로 데이터를 전개하는 방법도 배웠어. 정호, 너는 이 과정을 통해 무엇을 가장 중요하게 느꼈어?

정호: 나는 im2col 함수를 사용하는 부분이 특히 인상적이었어. 이 방법을 통해 합성곱 연산을 보다 효율적으로 수행할 수 있었지. 데이터를 전개하고 나서 합성곱 계층을 구현할 때, 계산 속도가 많이 빨라지는 걸 느꼈거든.

준형: 맞아, im2col을 사용해서 합성곱 연산을 행렬 곱셈으로 바꿔서 효율성을 높일 수 있었어. 그런데 여기서 합성곱 계층의 구현에서 주의해야 할 점은 뭐라고 생각해?

정호: 합성곱 계층을 구현할 때는 필터의 크기와 패딩, 스트라이드 설정이 중요하지. 이 값들에 따라 출력 크기가 크게 달라지니까. 특히 패딩과 스트라이드는 출력 특성 맵의 크기를 조절하는 데 중요한 역할을 해.

준형: 그렇지. 또 하나 중요한 것은 풀링 계층이야. 풀링 계층을 통해 데이터의 공간적 크기를 줄이고, 불필요한 정보나 잡음을 제거할 수 있었지. 풀링 계층을 구현할 때 가장 중요하다고 생각하는 부분은 뭐야?

정호: 풀링 계층에서는 주로 최대 풀링(max pooling)과 평균 풀링(average pooling)을 사용하잖아. 최대 풀링은 중요한 특징을 유지하면서 크기를 줄여주고, 평균 풀링은 전체적인 특징을 부드럽게 만들어주는 역할을 해. 그래서 풀링 방법을 선택하는 것도 중요하지.

준형: 풀링 방법의 선택이 모델의 성능에 미치는 영향이 크지. 그리고 우리는 SimpleConvNet을 통해 CNN을 구성하고, 가중치를 초기화하는 방법도 배웠어. CNN의 각 계층에서 어떤 정보가 추출되는지도 배웠고. 그럼 CNN에서 가중치 초기화가 중요한 이유는 뭐라고 생각해?

정호: 가중치 초기화는 신경망 학습에서 매우 중요해. 잘못된 초기값은 학습을 불안정하게 만들거나, 학습이 시작도 되기 전에 멈춰버릴 수도 있어. 우리가 배운 Xavier 초기화나 He 초기화는 이런 문제를 해결하기 위한 방법들이지. 이를 통해 가중치를 적절하게 초기화해서 학습이 원활하게 진행될 수 있도록 하는 거야.

준형: 맞아, 가중치 초기화는 신경망 학습의 시작점이니까. 그럼 이제 CNN의 실전 적용 예시를 한번 들어보자. 예를 들어, 우리가 이미 배운 AlexNet과 LeNet의 차이를 바탕으로 실제 애플리케이션에서 어떤 경우에 어떤 모델을 선택할지 고민해보자.

정호: LeNet은 비교적 간단한 구조로, 손글씨 인식 같은 간단한 문제에 적합해. 반면, AlexNet은 훨씬 더 깊고 복잡한 구조로, 이미지넷 같은 대규모 데이터셋에 적합하지. 그래서 문제의 복잡도와 데이터의 크기에 따라 모델을 선택하는 게 중요해.

준형: 그래, 맞아. 그래서 프로젝트를 시작할 때는 데이터셋과 문제의 특성을 잘 분석하고 적합한 모델을 선택하는 것이 중요해. 마지막으로, CNN 학습에서 자주 놓치는 부분 중 하나는 무엇일까?

정호: 나는 하이퍼파라미터 튜닝이 가장 어려운 부분이라고 생각해. 특히 학습률이 매우 중요한데, 학습률이 너무 크거나 작으면 학습이 제대로 이루어지지 않거든. 그리고 배치 정규화나 드롭아웃 같은 기법들도 신경 써야 해. 이 기법들은 학습 속도를 높이고, 오버피팅을 방지하는 데 큰 도움을 주지.

준형: 맞아, 하이퍼파라미터 튜닝은 많은 실험과 경험이 필요한 부분이지. 그럼 이제 우리는 배운 내용을 바탕으로 과제 #2를 하면서 우리가 test 정확도를 높이기위해 데이터셋의 범위를 정하지 않고, 모든 데이터셋을 사용하려고 했던 포인트도, 데이터 확장을 통한 정확도 높이기였던 것 같아. 설명을 돕기위해 아래에 과제#2를 해결하기전, 수업시간에 하면서 설정했던 데이터의 범위를 조절하는 이 코드를 담아봤어.

```
# 학습 데이터의 크기를 줄임 (데모 용도)

x_train, t_train = x_train[:5000], t_train[:5000]

x_test, t_test = x_test[:1000], t_test[:1000]
```

정호: 맞아. 나도 그 부분을 떠올린 너가 대단하다고 생각해. 이번 과제도 배운 내용을 통해 잘 해결할 수 있었어. 우리둘다 너무 고생많았어!! 마지막으로 우리 데이터셋을 사용하려한 포인트를 여기에 첨부해볼게.

정호: 예제에서 이 부분을 없애고, 과제#2에서는 데이터의 크기를 다 사용한게 아주 큰 포인트였던 것 같아.

8장

준형: 이번 8장에서 딥러닝의 특징과 정확도를 높이는 기술들을 배웠는데, 그중에서도 층을 깊게 쌓는 것이 모델의 성능을 향상시키는 데 큰 영향을 준다고 들었어. 그렇다면 층을 깊게 쌓을 때의 이점이 정확히 무엇인지 알려줄 수 있을까?

정호: 그래, 층을 깊게 쌓는 것이 딥러닝 모델의 성능을 향상시키는 이유는 여러 가지가 있는데, 주된 이유 중 하나는 특징의 계층적 추상화야. 층을 깊게 쌓으면 모델이 입력 데이터의 추상적인 특징을 학습할 수 있게 되는데, 예를 들어 낮은 층에서는 간단한 에지나 질감과 같은 저수준 특징을 학습하고, 이를 바탕으로 높은 층에서는 보다 추상적이고 의미 있는 특징을 학습할 수 있어. 이를 통해 모델이 더 복잡한 문제를 해결할 수 있게 되고, 정확도를 높일 수 있게 돼.

준형: 그렇구나!, 층을 깊게 쌓는 것이 모델의 추상화 능력을 향상시키는 데 중요하다는 거군. 그런데 딥러닝이 주목받게 된 계기가 무엇인지 궁금해.

정호: 딥러닝이 주목받게 된 계기는 주로 두 가지 이유가 있어. 첫 번째는 데이터의 증가와 컴퓨팅 파워의 발전이야. 과거에는 충분한 양의 데이터를 수집하고 처리하는 것이 어려웠는데, 인터넷과 모바일 기기의 보급으로 많은 양의 데이터를 수집하고 저장할 수 있게 되었고, GPU와 같은 고성능 컴퓨팅 자원이 저렴해지면서 대규모의 딥러닝 모델을 효율적으로 학습시킬 수 있게 되었어. 두 번째는 딥러닝의 효과적인 학습 알고리즘의 발전이야. 특히 역전파(backpropagation) 알고리즘의 발전과 함께 딥러닝 모델을 학습시키는 데 필요한 시간과 노력이 크게 줄어들었어.

정호: 데이터의 양과 컴퓨팅 파워의 발전이 딥러닝의 주목받게 된 계기였지. 그런데 딥러닝에서는 대량의 연산이 필요하다고 들었는데, 이 연산을 어떻게 처리하는지 준형이 너가 알려줄 수 있니?

준형: 그럼! 딥러닝에서 대량의 연산을 처리하는 데에는 주로 GPU 컴퓨팅이 활용돼. GPU는 병렬 처리가 가능하고 고속의 연산을 수행할 수 있어서 딥러닝 모델의 학습과 추론에 매우 효율적으로 사용해. 또한, 분산 학습(distributed learning)이라는 기술도 활용되는데, 이는 여러 대의 컴퓨터에 데이터를 분산시켜 학습을 동시에 수행함으로써 학습 속도를 높이고 대용량 데이터를 처리할 수 있게 돼. 이렇게 함으로써 딥러닝 모델이 더욱 복잡한 문제를 다룰 수 있게 되었어.

정호: 잘 알겠어. GPU 컴퓨팅과 분산 학습이 딥러닝에서 대량의 연산을 처리하는 데에 중요한 역할을 한다는 거군. 그런데 딥러닝에서 사용되는 하이퍼파라미터나 다양한 기법도 중요하다고 들었는데, 그 중에서도 학습률이 얼마나 중요한지 알려줘.

준형: 학습률은 딥러닝에서 매우 중요한 하이퍼파라미터 중 하나야. 학습률은 모델이 학습할 때 얼마나 큰 단계로 매개변수를 업데이트할지 결정하는 파라미터인데, 너무 크면 학습이 불안정해지고 발산할 수 있고, 너무 작으면 학습 속도가 너무 느려져서 최적의 모델을 찾을 수 없어. 따라서 적절한 학습률을 설정하는 것이 모델의 성능을 향상시키는 데에 중요한 역할을 해.

정호: 학습률이 모델의 학습에 직접적으로 영향을 미친다는 거군. 학습률을 적절히 조절하는 것이 모델의 학습에 있어서 결정적인 역할을 한다는 걸 알았어. 이렇게 여러 가지 기술과 하이퍼파라미터들이 상호작용하면서 딥러닝 모델의 성능을 향상시키는 걸 보면 정말 흥미로워. 앞으로 더 많은 것을 배워가면서 딥러닝 분야에서 더 나은 결과를 얻어내고 싶어.

준형: 나도 동의하는 의견이야. 딥러닝은 계속해서 발전하고 있으며, 새로운 기술과 방법들이 계속 등장하고 있어. 우리는 항상 새로운 것을 배우고 적용하여 모델을 개선해 나갈 필요가 있어. 함께 노력하면서 더 나은 결과를 얻을 수 있을거야!!!

정호: 고생많았어 이번학기 수업동안~! 방학 잘 보내고 2학기때 만나자.

준형: 그래! 방학 잘보내 정호야~

jupyter week13_0529 Last Checkpoint: 7 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[3]:
```
import numpy as np

x = np.random.rand(10, 1, 28, 28)
x.shape
```

[3]:
```
(10, 1, 28, 28)
```

[4]:
```
x[0].shape
```

[4]:
```
(1, 28, 28)
```

[5]:
```
x[0, 0]
```

[5]:
```
array([[0.91453549, 0.88384258, 0.59122711, 0.08847104, 0.58835591,
        0.4930357 , 0.56905157, 0.59563269, 0.46402532, 0.81661823,
        0.74583795, 0.95040402, 0.4071998 , 0.62007432, 0.32721347,
        0.00929344, 0.44763514, 0.76163058, 0.32543242, 0.36565056,
        0.31288352, 0.40899012, 0.4038817 , 0.30833356, 0.0459651 ,
        0.18082521, 0.60939051, 0.66483849],
       [0.67880703, 0.22779529, 0.1746541 , 0.54323404, 0.5610966 ,
        0.2577941 , 0.73504922, 0.11882056, 0.78745319, 0.76744899,
        0.06075351, 0.9486442 , 0.13735171, 0.58747826, 0.98803001,
        0.93317671, 0.62143588, 0.15876427, 0.80279943, 0.87196554,
        0.14096013, 0.60849594, 0.12474313, 0.71291497, 0.56725496,
        0.56509799, 0.20211993, 0.30187503],
       [0.74331229, 0.93153886, 0.14768637, 0.32707492, 0.3387561 ,
        0.0473211 , 0.35419118, 0.7250319 , 0.09861985, 0.68988442,
        0.30677847, 0.20349728, 0.98482637, 0.46044957, 0.9895353 ,
        0.56042376, 0.476544  , 0.5133625 , 0.81437285, 0.67577866,
        0.77631989, 0.68666253, 0.48252346, 0.49875356, 0.69265662,
        0.13625591, 0.15954017, 0.79940637]
```

jupyter week13_0529 Last Checkpoint: 7 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[6]:
```
import sys, os
import numpy as np
sys.path.append(os.pardir)
from common.util import im2col

x1 = np.random.rand(1, 3, 7, 7)
col1 = im2col(x1, 5, 5, stride=1, pad=0)
print(col1.shape)

x2 = np.random.rand(10, 3, 7, 7)
col2 = im2col(x2, 5, 5, stride=1, pad=0)
print(col2.shape)
```

```
(9, 75)
(90, 75)
```

[7]:
```
class Pooling:
    def __init__(self, pool_h, poo_w, stride=1, pad=0):
        self.pool_h = pool_h
        self.pool_w = pool_w
        self.stride = stride
        self.pad = pad

    def forward(self, x):
        N, C, H, W = x.shape
        out_h = int(1 + (H - self.pool_h) / self.stride)
        out_w = int(1 + (W - self.pool_w) / self.stride)

        # (1) 전개
        col = im2col(x, self.pool_h, self.pool_w, self.stride, self.pad)
        col = col.reshape(-1, self.pool_h*self.pool_w)

        # (2) 최댓값
```

jupyter week13_0529 Last Checkpoint: 7 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[7]:
```python
class Pooling:
    def __init__(self, pool_h, poo_w, stride=1, pad=0):
        self.pool_h = pool_h
        self.pool_w = pool_w
        self.stride = stride
        self.pad = pad

    def forward(self, x):
        N, C, H, W = x.shape
        out_h = int(1 + (H - self.pool_h) / self.stride)
        out_w = int(1 + (W - self.pool_w) / self.stride)

        # (1) 전개
        col = im2col(x, self.pool_h, self.pool_w, self.stride, self.pad)
        col = col.reshape(-1, self.pool_h*self.pool_w)

        # (2) 최댓값
        out = np.max(col, axis=1)
        # (3) 성형
        out = out.reshape(N, out_h, out_w, C).transpose(0, 3, 1, 2)

        return out
```

jupyter assightment+SimpleConvNet Last Checkpoint: 7 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[ ]:
```python
import sys
import os
sys.path.append(os.pardir)  # 부모 디렉토리의 파일을 가져오기 위한 설정
import numpy as np
import matplotlib.pyplot as plt
from dataset.mnist import load_mnist
from common.trainer import Trainer
from collections import OrderedDict
from common.layers import *
from common.gradient import numerical_gradient

class SimpleConvNet:
    """
    다음과 같은 CNN을 구성한다.
    → Conv → ReLU → Pooling → Affine → ReLU → Affine → Softmax →
    """
    def __init__(self, input_dim=(1, 28, 28),
                 conv_param={'filter_num': 30, 'filter_size': 5, 'pad': 0, 'stride': 1},
                 hidden_size=100, output_size=10, weight_init_std=0.01):
        filter_num = conv_param['filter_num']
        filter_size = conv_param['filter_size']
        filter_pad = conv_param['pad']
        filter_stride = conv_param['stride']
        input_size = input_dim[1]
        conv_output_size = (input_size - filter_size + 2*filter_pad) / filter_stride + 1
        pool_output_size = int(filter_num * (conv_output_size/2) * (conv_output_size/2))

        # 가중치 초기화
        self.params = {}
        self.params['W1'] = weight_init_std * np.random.randn(filter_num, input_dim[0], filter_size, filter_size)
```

```
        # 가중치 초기화
        self.params = {}
        self.params['W1'] = weight_init_std * np.random.randn(filter_num, input_dim[0], filter_size, filter_size)
        self.params['b1'] = np.zeros(filter_num)
        self.params['W2'] = weight_init_std * np.random.randn(pool_output_size, hidden_size)
        self.params['b2'] = np.zeros(hidden_size)
        self.params['W3'] = weight_init_std * np.random.randn(hidden_size, output_size)
        self.params['b3'] = np.zeros(output_size)

        # 계층 생성
        self.layers = OrderedDict()
        self.layers['Conv1'] = Convolution(self.params['W1'], self.params['b1'], conv_param['stride'], conv_param['pad'])
        self.layers['Relu1'] = Relu()
        self.layers['Pool1'] = Pooling(pool_h=2, pool_w=2, stride=2)
        self.layers['Affine1'] = Affine(self.params['W2'], self.params['b2'])
        self.layers['Relu2'] = Relu()
        self.layers['Affine2'] = Affine(self.params['W3'], self.params['b3'])
        self.last_layer = SoftmaxWithLoss()

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

    def loss(self, x, t):
        y = self.predict(x)
        return self.last_layer.forward(y, t)

    def accuracy(self, x, t, batch_size=100):
        if t.ndim != 1:
            t = np.argmax(t, axis=1)

        acc = 0.0
```

jupyter assightment+SimpleConvNet Last Checkpoint: 7 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```
        for i in range(int(x.shape[0] / batch_size)):
            tx = x[i*batch_size:(i+1)*batch_size]
            tt = t[i*batch_size:(i+1)*batch_size]
            y = self.predict(tx)
            y = np.argmax(y, axis=1)
            acc += np.sum(y == tt)

        return acc / x.shape[0]

    def gradient(self, x, t):
        self.loss(x, t)
        dout = 1
        dout = self.last_layer.backward(dout)

        layers = list(self.layers.values())
        layers.reverse()
        for layer in layers:
            dout = layer.backward(dout)

        grads = {}
        grads['W1'] = self.layers['Conv1'].dW
        grads['b1'] = self.layers['Conv1'].db
        grads['W2'] = self.layers['Affine1'].dW
        grads['b2'] = self.layers['Affine1'].db
        grads['W3'] = self.layers['Affine2'].dW
        grads['b3'] = self.layers['Affine2'].db

        return grads

# 데이터 로드
(x_train, t_train), (x_test, t_test) = load_mnist(flatten=False)

# 학습 데이터의 크기를 줄임 (데모 용도)
```

jupyter assightment+SimpleConvNet Last Checkpoint: 7 days ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

```
x_train, t_train = x_train[:5000], t_train[:5000]
x_test, t_test = x_test[:1000], t_test[:1000]

# SimpleConvNet을 초기화
network = SimpleConvNet(input_dim=(1, 28, 28),
                        conv_param={'filter_num': 30, 'filter_size': 5, 'pad': 0, 'stride': 1},
                        hidden_size=100, output_size=10, weight_init_std=0.01)

# Trainer로 학습
trainer = Trainer(network, x_train, t_train, x_test, t_test,
                  epochs=100, mini_batch_size=100,
                  optimizer='Adam', optimizer_param={'lr': 0.001},
                  evaluate_sample_num_per_epoch=1000)
trainer.train()

# 그래프 그리기
markers = {'train': 'o', 'test': 's'}
x = np.arange(20)
plt.plot(x, trainer.train_acc_list, marker='o', label='train', markevery=2)
plt.plot(x, trainer.test_acc_list, marker='s', label='test', markevery=2)
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.ylim(0, 1.0)
plt.legend(loc='lower right')
plt.show()
```

```
train loss:2.2999469358191504
=== epoch:1, train acc:0.243, test acc:0.252 ===
train loss:2.296260356951048
train loss:2.296567968007572
train loss:2.2905454233602307
train loss:2.280136640218573
train loss:2.269715021736389
```

```
train loss:0.002120067451615632
train loss:0.002434757503340098
train loss:0.0045745223344921485
train loss:0.0015743221552548753
train loss:0.00545221449726934
train loss:0.0038648243742702886
=== epoch:33, train acc:0.999, test acc:0.956 ===
train loss:0.0014210896809569095
train loss:0.0031774505325306347
train loss:0.0056452569634140295
train loss:0.004449537508533851
train loss:0.005660869127164166
train loss:0.001610494102057172
train loss:0.0018956706433875688
train loss:0.0015125953553892652
train loss:0.002139598540618564
train loss:0.002950699700287767
train loss:0.0018327358705901467
train loss:0.002011267380649660
```

[ ]: