인공지능 정리 노트 4

소프트웨어 202284011 김연재

6.4 바른 학습을 위해:

오버피팅 문제: 모델이 훈련 데이터에 지나치게 적응하여 새로운 데이터에 대응하지 못하는 현상.

가중치 감소와 드롭아웃 기술을 이용해 오버피팅을 억제한다.

6.4.1 오버피팅:

매개변수가 많고 복잡한 모델 또는 적은 훈련 데이터를 가진 모델에서 나타난다.

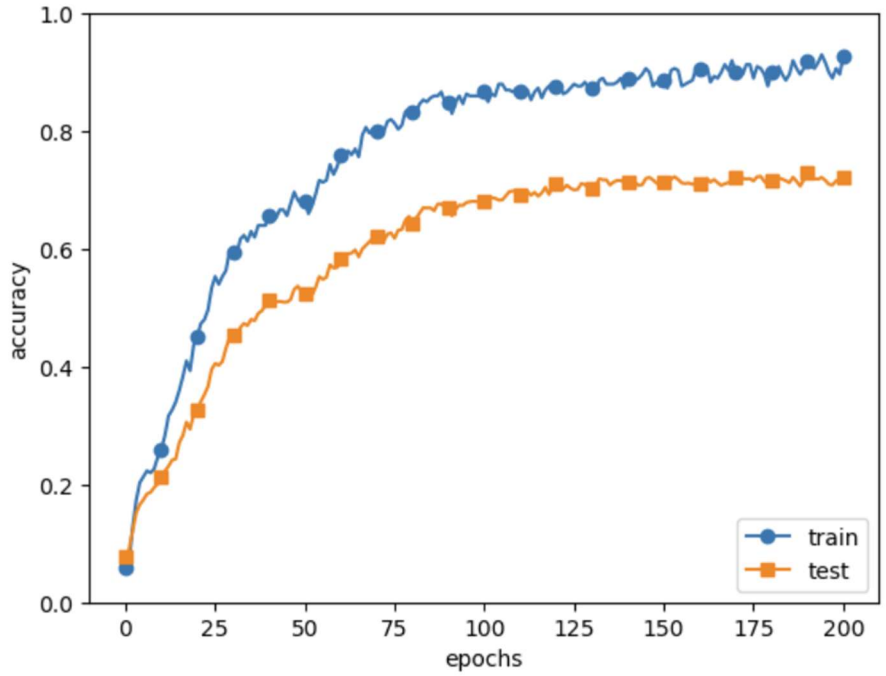
6.4.2 가중치 감소:

손실 함수에 가중치의 L2 노름 추가한다. 큰 가중치에 대해서는 페널티를 부과하여 오버피팅 억제하는 방법,

코드

```
network = MultiLayerNet(input_size=784, hidden_size_list=[100, 100, 100, 100, 100, 100], output_size=10,
                        weight_decay_lambda=weight_decay_lambda)
optimizer = SGD(lr=0.01) # 학습률이 0.01인 SGD로 매개변수 갱신

max_epochs = 201
train_size = x_train.shape[0]
batch_size = 100

train_loss_list = []
train_acc_list = []
test_acc_list = []

iter_per_epoch = max(train_size / batch_size, 1)
epoch_cnt = 0

for i in range(1000000000):
    batch_mask = np.random.choice(train_size, batch_size)
    x_batch = x_train[batch_mask]
    t_batch = t_train[batch_mask]

    grads = network.gradient(x_batch, t_batch)
    optimizer.update(network.params, grads)

    if i % iter_per_epoch == 0:
        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("epoch:" + str(epoch_cnt) + ", train acc:" + str(train_acc) + ", test acc:" + str(test_acc))

        epoch_cnt += 1
        if epoch_cnt >= max_epochs:
            break
```

실험 결과



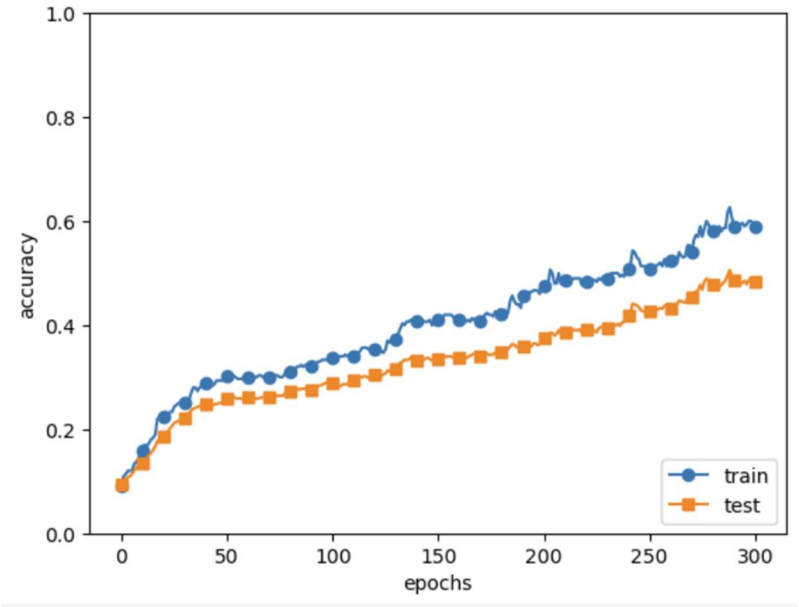
6.4.3 드롭아웃:

신경망 모델이 복잡해지만 가중치 감소만으로 개응하기 어려워 드롭아웃 기법을 이용한다. 훈련 시 뉴런을 임의로 삭제한다. 모델 복잡성을 높이면서 오버피팅을 억제할 수 있다.

코드

```
network = MultiLayerNetExtend(input_size=784, hidden_size_list=[100, 100, 100, 100, 100, 100],
                              output_size=10, use_dropout=use_dropout, dropout_ration=dropout_ratio)
trainer = Trainer(network, x_train, t_train, x_test, t_test,
                  epochs=301, mini_batch_size=100,
                  optimizer='sgd', optimizer_param={'lr': 0.01}, verbose=True)
trainer.train()

train_acc_list, test_acc_list = trainer.train_acc_list, trainer.test_acc_list
```

실험 결과



6.5 적절한 하이퍼파라미터 값 찾기:

검증 데이터를 사용해 하이퍼파라미터 조정한다.

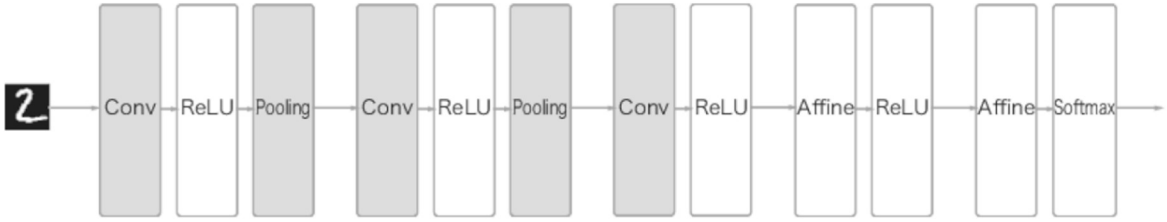
최적화 단계: 값 범위 설정, 무작위 추출, 정확도 평가 및 범위 축소.

7. 합성곱 신경망 (CNN):

이미지 인식 및 음성 인식에서 주로 사용한다. 국부수용장 모델을 모방한 구조다.

7.1 전체 구조:

CNN은 합성곱 계층과 풀링 계층을 포함한다. 마지막 출력 계층에서 Affine – Softmax 조합을 사용한다.



7.2 합성곱 계층:

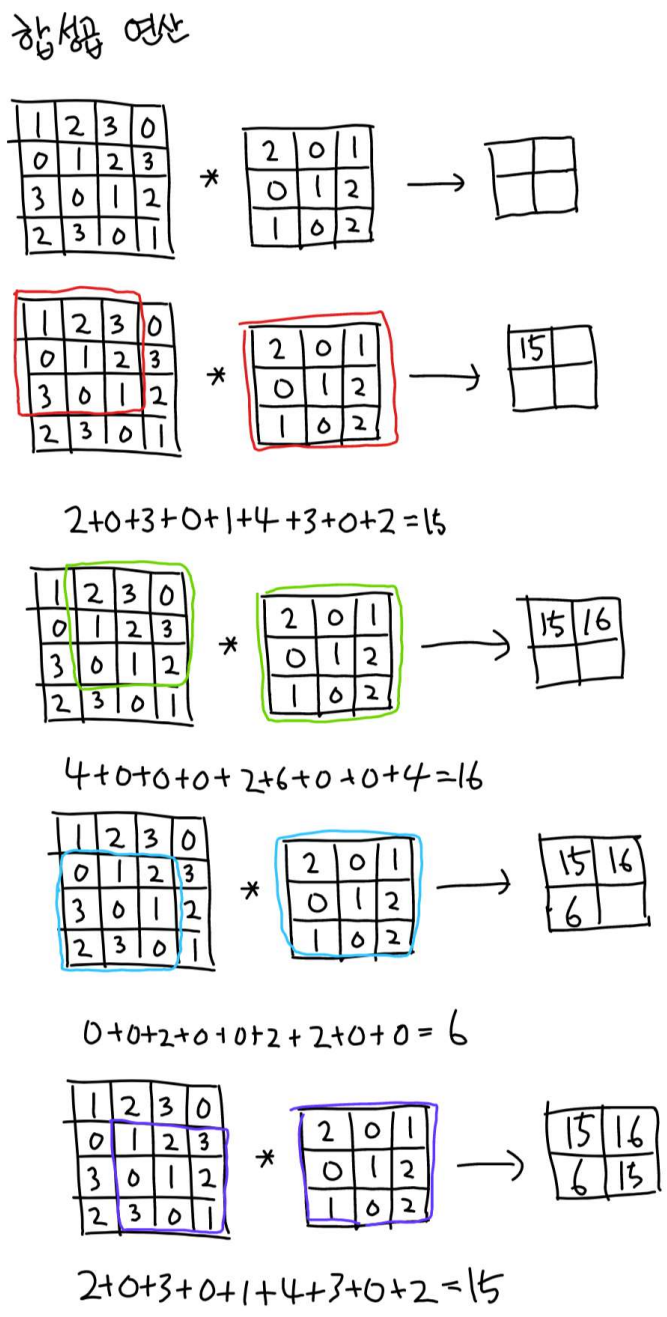
합성곱 연산: 입력 데이터에 필터를 적용하여 특징 맵을 생성. 패딩과 스트라이드 등의 개념 사용한다.

7.2.1 완전연결 계층의 문제점:

데이터 형상을 무시하고, 계산량이 많다.

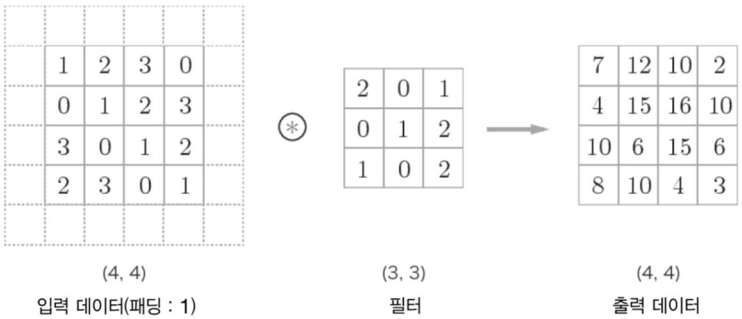
7.2.2 합성곱 연산:

필터를 입력 데이터에 적용하여 새로운 출력을 생성한다.



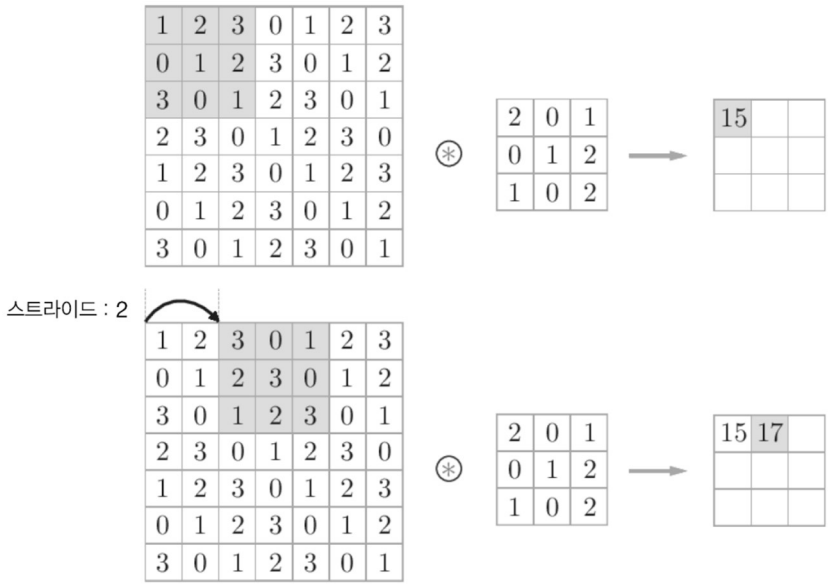
7.2.3 패딩

출력의 크기를 조정할 목적으로 사용한다. 합성곱 연산을 수행하기 전에 입력 데이터 주변을 특정 값으로 채운다.



7.2.4 스트라이드

필터를 적용하는 위치의 간격을 지정한다.



예제 1) 입력크기 = (4,4), 패딩 = 1, 스트라이드=1, 필터 = (3,3)일 때 출력의 크기는?

OH = ((4+2-3)/1)+1 = 4, OW = 4

예제 2) 입력크기 = (7,7), 패딩 = 0, 스트라이드=2, 필터 = (3,3)일 때 출력의 크기는?

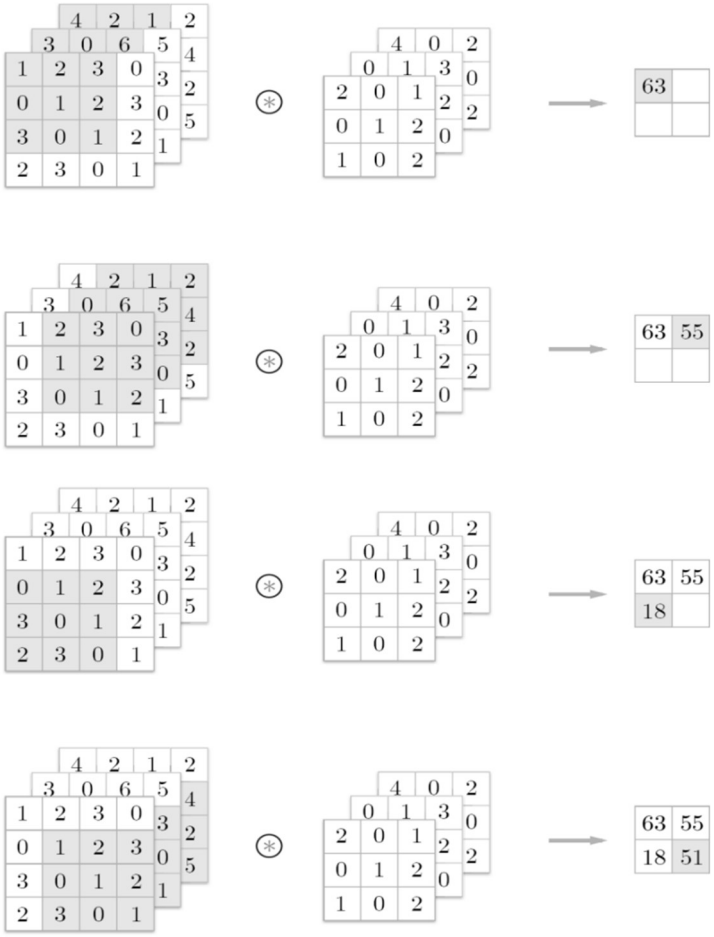
OH = ((7+0-3)/2)+1 = 3, OW = 3

예제 3) 입력크기 = (28,31), 패딩 = 2, 스트라이드=3, 필터 = (5,5)일 때 출력의 크기는?

OH = ((28+4-5)/3)+1 = 10, OW = ((31+4-5)/3)+1) = 11

7.2.5 3차원 데이터의 합성곱 연산

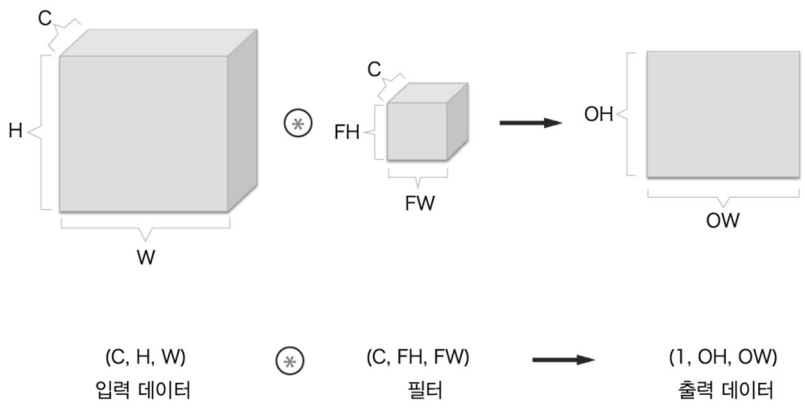
이미지는 세로, 가로, 채널인 3차원 데이터이다.



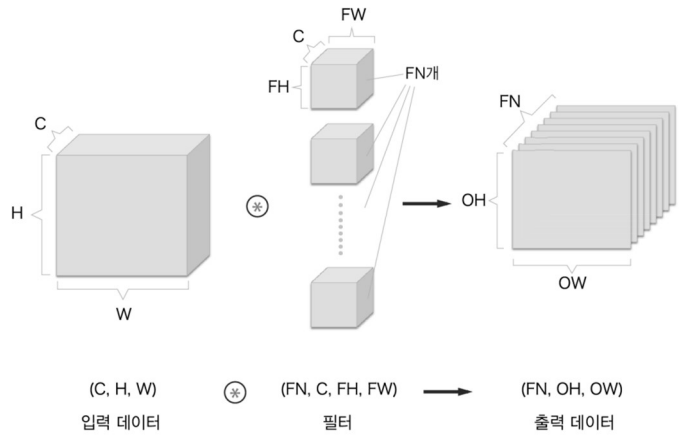
입력 데이터의 채널 수와 필터의 채널 수가 같아야 함. 모든 채널의 필터는 같은 크기여야 함.

7.2.6 블록으로 생각하기

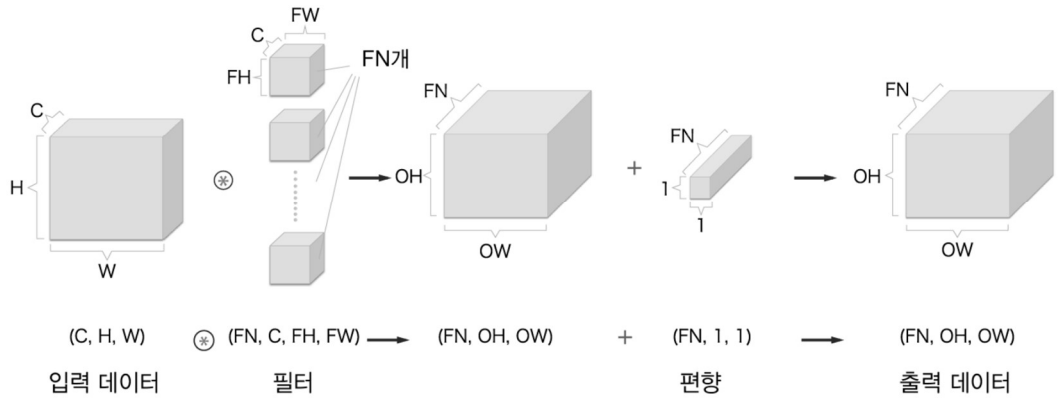
3차원 데이터를 다차원 배열로 나타낼 때 채널(C), 높이(H), 배열(W) 순서로 사용함.



필터를 FN개 적용하면 출력맵도 FN개가 생성됨. FN개의 맵을 모으면 형상이 FN, OH, OW인 블록이 완성된다. 필터의 가중치 데이터는 출력 채널 수, 입력 채널 수, 높이, 너비로 이루어진 4차원 데이터이다.



합성곱 연산의 편향 – 편향은 채널 하나에 값 하나씩 구성된다. 편향의 형상은 FN, 1, 1이고 필터의 출력 결과의 형상은 FN, OH, OW 이다.

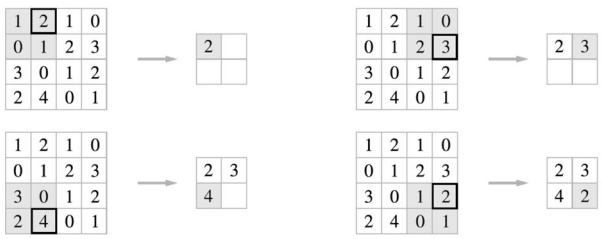


7.2.7 배치처리

배치 처리 시의 데이터 흐름을 보면 각 데이터의 선두에 배치용 차원을 추가한다. 데이터는 4차원 영상을 가지고 각 계층을 타고 흐른다. 신경망에 4차원 데이터가 하나 흐를 때마다 데이터 N개에 대한 합성곱 연산이 이루어진다.

7.3 풀링 계층

입력 데이터의 크기를 줄이고 계산 효율성을 높이며 모델의 성능을 향상시킨다.



7.3.1 풀링 계층의 특징

학습해야 할 매개변수가 없다. -> 처리할 것도 없고 편향이 없다. 그저 큰 값을 고른다.

채널의 수가 변하지 않는다. 입력의 변화에 영향을 적게 받는다.

6장 질문

1) Q. 교차 검증을 사용한 하이퍼파라미터 조정 과정이 어떻게 되는가? 이 방법이 모델 성능을 향상시키는 방식은 무엇인가?

   A. 교차 검증을 사용한 하이퍼파라미터 조정은 데이터를 여러 개의 폴드로 나누어 각 폴드에서 다양한 하이퍼파라미터 조합의 성능을 평가하는 과정이다. 각 조합은 여러 폴드에서 테스트되며, 평균 성능이 가장 좋은 하이퍼파라미터를 선택한다. 이 방법은 특정 데이터셋에 과적합되지 않도록 하여 일반화 성능을 향상시킨다.

2) Q. 하이퍼파라미터 최적화에서 그리드 서치와 랜덤 서치 중 어떤게 더 좋은가?

   A. 그리드 서치는 미리 정의된 하이퍼파라미터 값을 체계적으로 탐색하여 모든 조합을 테스트한다. 철저하게 탐색하지만, 하이퍼파라미터나 값의 수가 많아지면 계산 비용이 크게 증가한다. 반면 랜덤 서치는 하이퍼파라미터 값을 무작위로 샘플링하여 효율적으로 탐색할 수 있으며, 적은 평가 횟수로 좋은 솔루션을 찾을 수 있다. 그러나 잘 샘플링되지 않으면 최적의 조합을 놓칠 수도 있다.

7장 질문

1) Q. 합성곱 계층에서 필터 수(FN)의 중요성은? 필터 수가 학습된 특징과 전체 모델 복잡성에 미치는 영향은?

   A. 합성곱 계층에서 필터 수(FN)는 다양한 필터를 입력 데이터에 적용하여 생성된 특징 맵의 수를 결정한다. 필터 수가 많을수록 네트워크는 더 다양한 특징을 학습할 수 있어 복잡한 패턴을 더 잘 포착할 수 있다. 그러나 필터 수가 많아지면 파라미터 수와 계산 복잡도가 증가하여 과적합 위험이 커지고 학습 시간이 길어질 수 있다.

2) Q. 맥스 풀링과 평균 풀링의 차이점은? 각 기법이 선호되는 시나리오는 어떤 경우인가?

   A. 맥스 풀링은 각 풀링 창에서 최대값을 선택하여 가장 두드러진 특징을 강조하고, 이동 불변성을 제공한다. 평균 풀링은 창 내의 평균값을 계산하여 특징의 일반적인 요약을 제공한다. 맥스 풀링은 강력한 특징 추출이 필요한 작업에서 자주 사용되며, 평균 풀링은 특징의 전반적인 경향을 보존하고자 할 때 유용하다.