

실습문제: 6장. 학습 관련 기술들(6.4, 6.5)

학번: 202404178 이름: 강우현

1. Overfitting

(1) 모델이 오버피팅되는 두 가지 경우는?

매개변수 수가 많고 복잡한 경우, 훈련 데이터가 매우 적은 경우입니다.

(2) [그림 6-20] 그래프를 보고 오버피팅이라고 말할 수 있는 이유는?

훈련 정확도는 거의 100%로 계속 상승하지만 테스트 정확도는 하락하기 때문입니다.

(3) [그림 6-21] 그래프가 오버피팅이 아니고 보는 이유는?

언더피팅일 경우 혹은 학습률 설정 때문에 충분히 학습되지 않은 상황일 확률이 높기 때문입니다.

2. Weight decay

(1) 가중치 감소 무엇이고, 어떻게 구현하는가? 문장으로 설명하시오.

오버피팅을 줄이는 기법입니다. 일반적으로 손실 함수에 가중치의 제곱합을 추가하여 구현합니다.

(2) [그림 6-21] 그래프가 오버피팅이 아니라고 보는 이유는? 정확도가 낮은 이유는 무엇일까?

가중치 크기를 억제해 모델의 표현력이 일부 감소하기 때문에 오버피팅이 아니라고 봅니다. 정확도가 낮은 이유는 오버피팅 억제는 성공했을 수 있지만 훈련에서도 높은 성능을 내지 못하였기 때문입니다.

(3) multi_layer_net.py 파일에서 Weight decay가 구현된 부분을 찾아 캡처하여 넣으시오.

```
weight_decay = 0
for idx in range(1, self.hidden_layer_num + 2):
    W = self.params['W' + str(idx)]
    weight_decay += 0.5 * self.weight_decay_lambda * np.sum(W ** 2)
```

3. dropout

(1) 드롭아웃을 하는 이유는? 드롭아웃을 어떻게 구현하는가?

오버피팅을 억제하고 일반화 성능을 향상시키기 위해 사용합니다.

구현 : 각 순전파에서 확률 dropout_ratio로 유닛을 제거하고, 제거하지 않은 유닛들만 다음 층으로 전달. 보통 내부적으로 mask = (np.random.rand(*x.shape) > dropout_ratio)를 만들고 out = x * mask 처리.

(2) common/layers.py 파일에서 테스트(시험)시 드롭아웃의 출력 코드를 찾아 적으시오. 이 값을 곱하는 이유를 추측해 보시오.

class Dropout:

"""

<http://arxiv.org/abs/1207.0580>

"""

def __init__(self, dropout_ratio=0.5):

self.dropout_ratio = dropout_ratio

```
self.mask = None
```

```
def forward(self, x, train_flg=True):  
    if train_flg:  
        self.mask = np.random.rand(*x.shape) > self.dropout_ratio  
        return x * self.mask  
    else:  
        return x * (1.0 - self.dropout_ratio)  
  
def backward(self, dout):  
    return dout * self.mask
```

곱하는 이유 : 각 뉴런의 출력 기대값을 학습 시와 동일하게 맞추기 위해

(3) dropout_ratio = 0.1이면, Dropout 계층에서 학습시 대략 몇 %의 노드를 제외시키는가?
약 10%의 노드

4. 최적의 하이퍼파라미터 결정

(1) 하이퍼파라미터와 파라미터는 각각 무엇을 의미하는가?

파라미터 : 학습으로부터 학습되는 값들

하이퍼파라미터 : 학습 전에 사람이 설정하는 값들

(2) 검증 데이터(Validation data)란?

하이퍼파라미터를 선택할 때 사용하는 데이터이다.

(3) 6.5.2를 참고하여 최적(최선)의 하이퍼파라미터를 결정(선택)하는 절차(4단계)를 설명하시오.

1단계 : 하이퍼파라미터의 탐색 범위를 정한다.

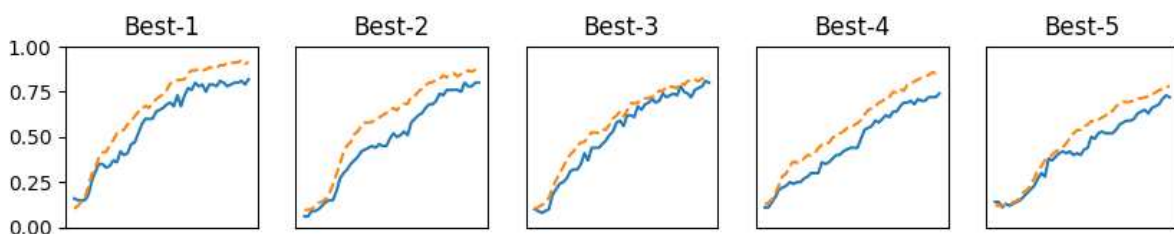
2단계 : 설정한 범위에서 하이퍼파라미터 값을 무작위로 샘플링한다.

3단계 : 샘플링한 하이퍼파라미터로 모델을 학습하고, 검증 데이터로 정확도 평가.

4단계 : 1~2단계를 여러 번 반복하여 결과를 모은 뒤, 성능이 좋았던 값 주변으로 범위를 좁혀 다시 반복

※ 아래 (4)-(6)은 ch06/hyperparameter_optimization.py 코드를 보고 물음에 답하시오.

(4) 6.5.3에서 두 하이퍼파라미터(학습률, 가중치감소 계수)를 최적화하였다. hyperparameter_optimization.py 파일을 각자 실행하여, p.226과 같이 Best-1부터 Best-5까지 결과를 캡처하여 붙여 넣으시오.



(5) 위 (4) 이후 하이퍼파라미터의 범위를 좁혀야 한다. 범위를 어떻게 좁히겠는가? (힌트: 10 **

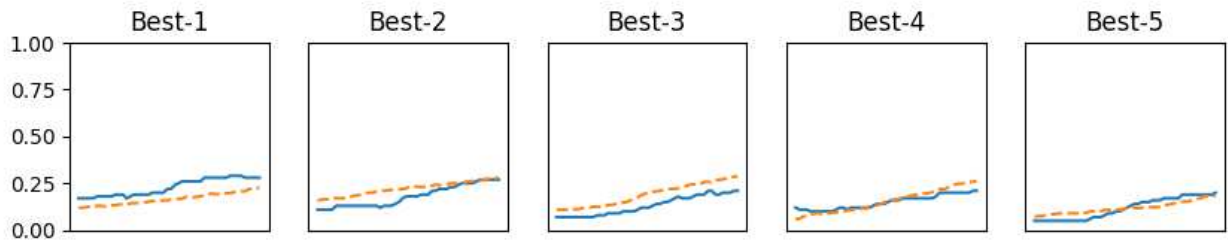
np.random.uniform(a, b) 대신 np.random.uniform(a, b)을 이용하면 더 쉽게 좁힐 수 있다)

초기에는 넓은 로그 스케일로 탐색하고 상위 성능 조합의 분포를 확인해서 좁힌 범위에서 다시 탐색을 진행할 것입니다.

(6) 위 (5)에서 좁힌 범위에 대해 위 (5)를 수행하여 최적의 파라미터를 결정하시오. (힌트: 최적의 파라미터를 찾기 위해 몇 번 더 범위를 좁혀 보는 것이 좋다)

```
weight_decay = 10 ** np.random.uniform(-6.5, -5)
```

```
lr = 10 ** np.random.uniform(-4.5, -3)
```



※ 템플시 이런 식의 하이퍼파라미터 최적화를 꼭 수행하시기 바랍니다.