[Week6]_문가을

딥러닝 2단계 : 심층 신경망 성능 향상시키기

1. 머신러닝 어플리케이션 설정하기

Train/Dev/Test 세트

- 딥러닝은 여러 분야에 사용되지만, 어떤 분야나 애플리케이션의 직관이 다른 애플리케이션 영역에 거의 적용되지 않음.
- 데이터의 양이나 입력 특성의 개수, 훈련을 진행하는 컴퓨터 설정 등 다양한 요인에 의해 결정됨.

→ Idea → code → experiment 사이클을 여러번 거치면서 최적의 하이퍼파라미터 찾기.

Train/Dev/Test 세트

70/30 or 60/20/20 ← 전통적인 비율

백만 개 이상의 데이터가 있을 때는 개발 세트와 테스트 세트의 비율이 매우 작아짐. → 평가할 수 있을 정도의 크기이기만 하면 됨.

train test 데이터의 distribution이 맞지 않을 수 있음

→ 같은 분포에서 와야 함.

test set가 필요하지 않을 수 있음.

테스트 세트의 목표는 최종 네트워크의 성능에 대한 비편향 추정을 제공하는 것임. → 비편향 추정이 필요 없는 경우에 테스트 세트가 필요하지 않음.

편향 / 분산

편향과 분산은 trade-off 관계임.

> 예측값들과 정답이 대체로 멀리 떨어져 있으면 결과의 편향(bias)이 높다고 말하고,
> 예측값들이 자기들끼리 대체로 멀리 흩어져있으면 결과의 분산(variance)이 높다고 말합니다.

참조 : https://www.opentutorials.org/module/3653/22071

높은 편향(high bias)	과소적합(underfitting)
알맞음(just right)	-
높은 분산(high variance)	과대적합(overfitting)

- 훈련 세트와 개발 세트의 관계

인간의 오류가 0%라고 가정했을 때 (Optimal (Bayes) error가 0%)

. 1) 2) 3) 4)

	높은 분산 (과대적합)	높은 편향 (과소적합)	높은 편향 & 높은 분산	낮은 편향 & 낮은 분산
훈련 세트	1 %	15 %	15 %	0.5 %
개발 세트	11 %	11 %	30 %	1 %

1) 알고리즘의 분산이 높음

2) 훈련 세트에서도 잘 작동하지 않음. → 과소적합 → 높은 편향

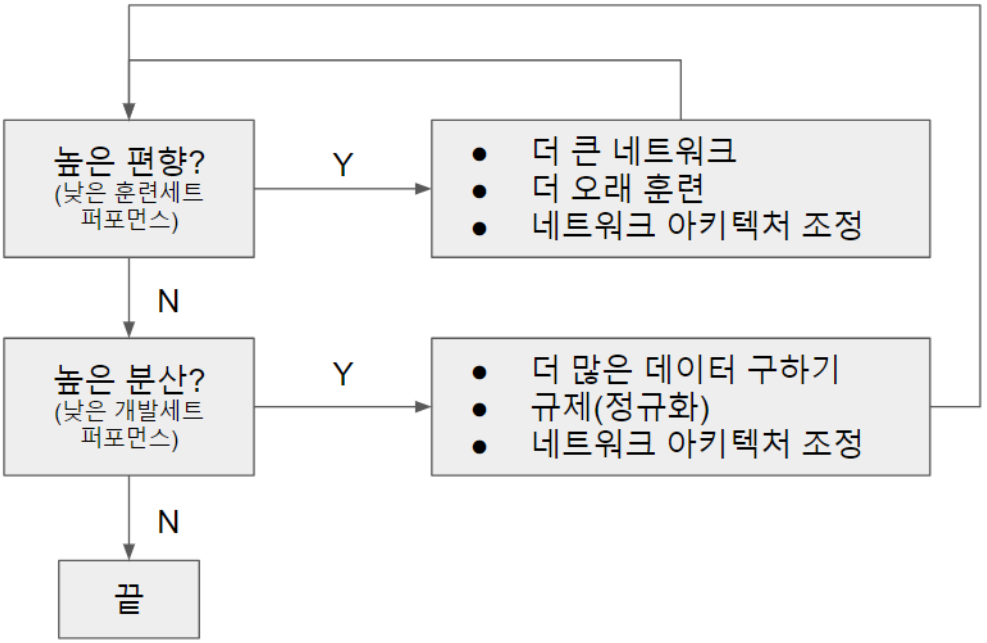
3) 훈련 세트와 맞지 않음 → 높은 편향

개발 세트에서도 맞지 않음 → 높은 분산

- 훈련세트에서 얼마나 알고리즘이 적합한지에 따라 편향 문제가 있는지 확인
- 훈련세트에서 개발 세트로 갈 때 오차가 얼마나 커지는지에 따라서 분산 문제 확인.

머신러닝을 위한 기본 레시피

알고리즘의 성능을 체계적으로 개선하기 위한 레시피!



1) 높은 편향이냐 분산이냐에 따라 시도해볼 수 있는 방법이 매우 달라질 수 있음.

2) 초기 머신러닝 시대에는 편향-분산 트레이드오프에 대한 많은 논의가 있었음. 서로를 나쁘게 하지 않고 어느 하나만 감소시키는 툴이 많이 없었기 때문임.

→ 현대의 딥러닝 빅데이터 시대에는 둘 중 하나만 감소시키는 툴 존재.

---

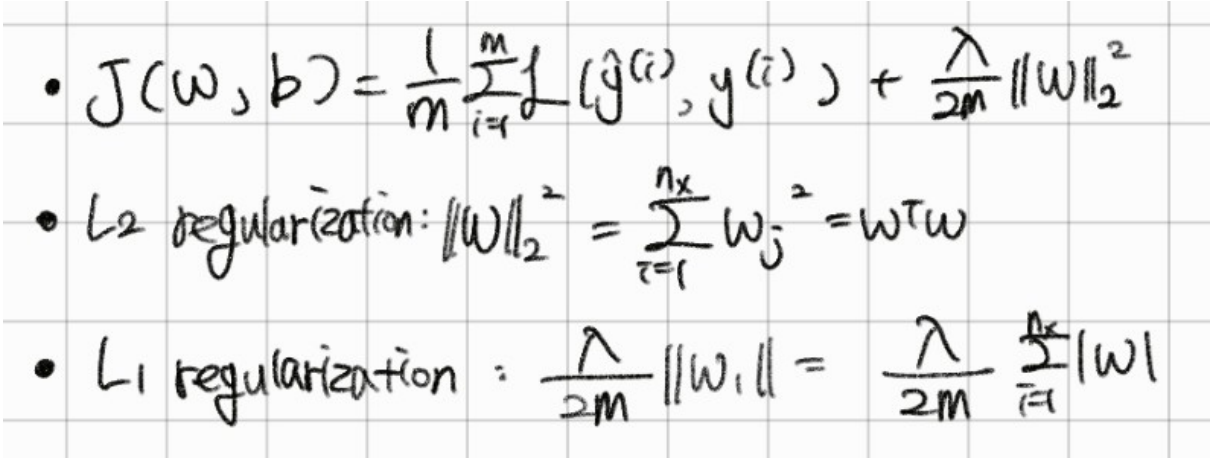
2. 신경망 네트워크의 정규화

정규화

높은 분산으로 신경망이 데이터를 과대적합하는 문제가 의심되면 정규화 시도!

→ 과대적합을 막고 신경망의 분산을 줄이는 데에 도움이 됨.

<로지스틱 회귀>



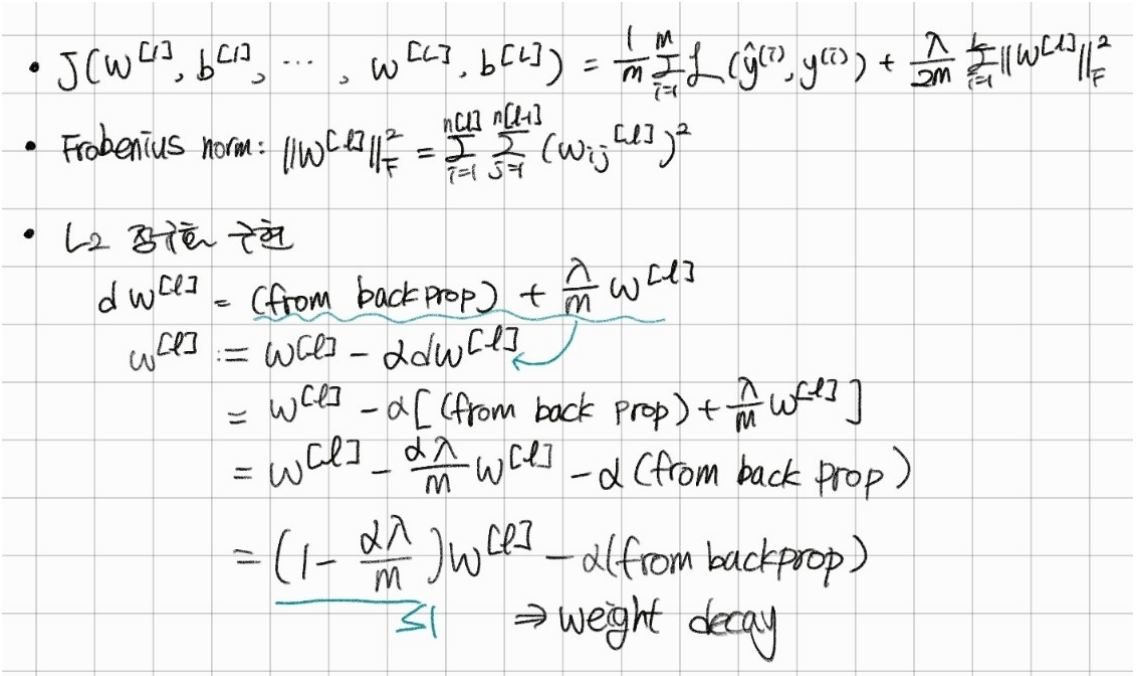
b 에 대한 정규화는 하지 않는 이유

→ w가 높은 차원의 매개 변수 벡터이고 b는 하나의 숫자이기 때문에, 거의 모든 매개변수는 b가 아닌 w에 있음. b에 대한 항을 추가해도 실질적인 차이 없음.

L1 regularization을 사용하게 되면

→ w 벡터 안에 0이 많아짐.

<Neural network>

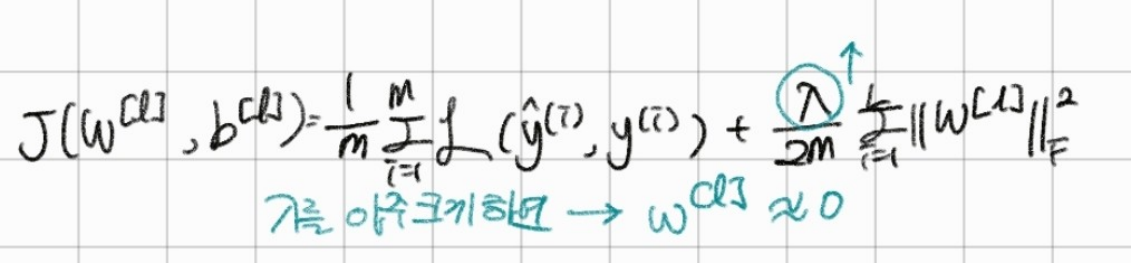


Weight decay 가중치 감소

왜 정규화는 과대적합을 줄일 수 있을까요?

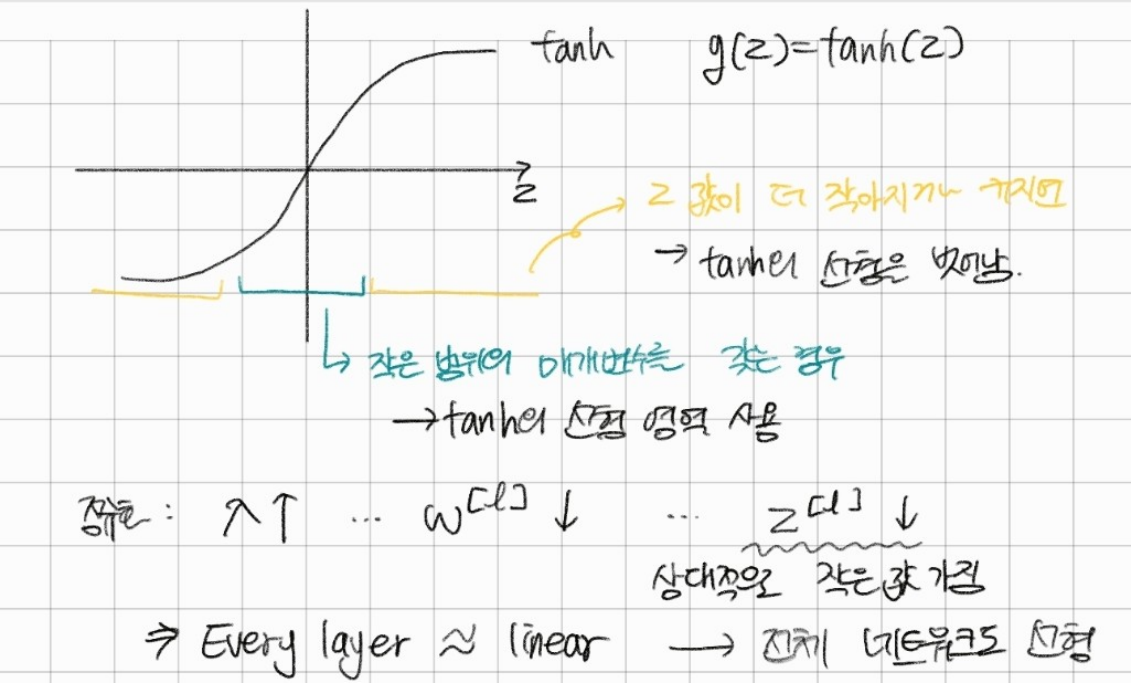
<L2 정규화>

직관 1 :



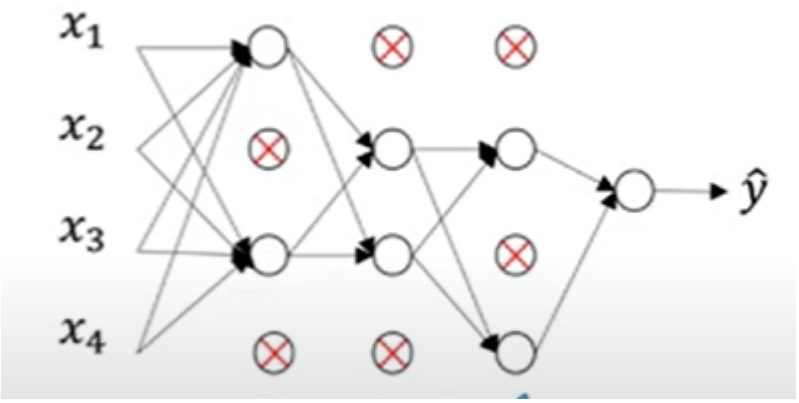
가중치가 0에 가까워지면 은닉 유닛의 영향력을 0에 가깝게 줄임으로써 네트워크가 간단해짐.

직관 2 :

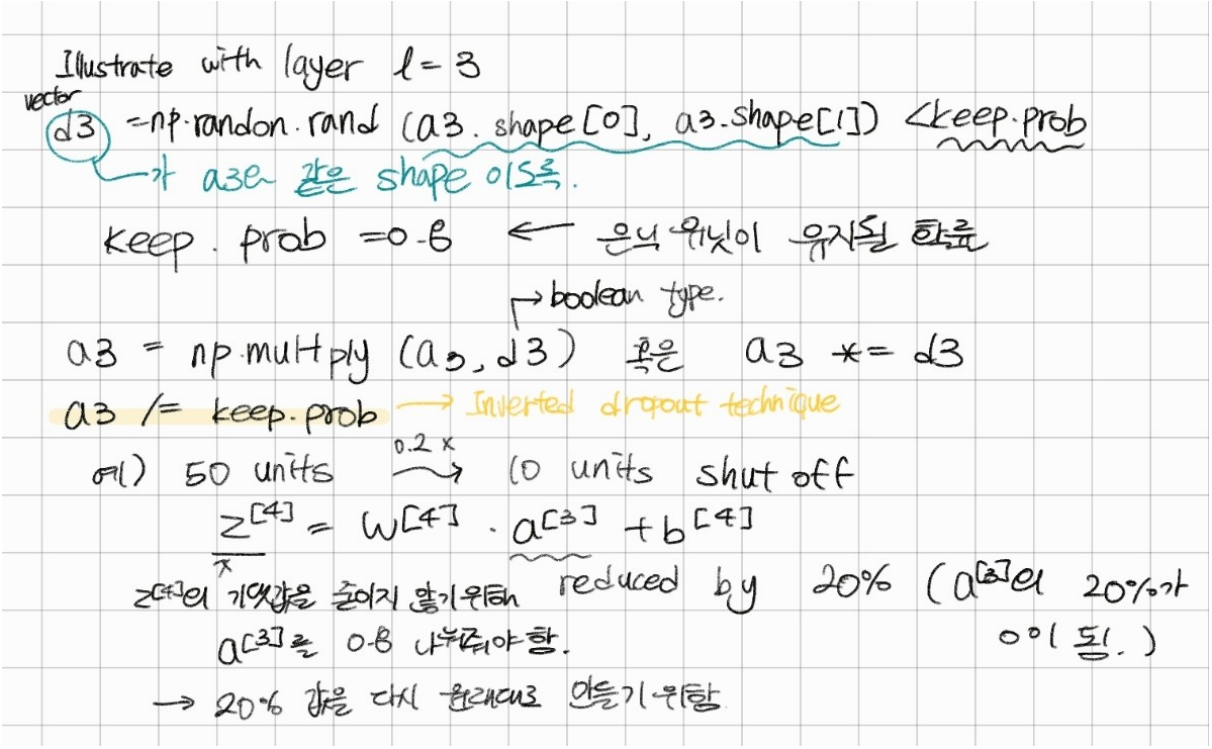


드롭아웃 정규화

드롭아웃의 방식은 신경망의 각각의 층에 대해 노드를 삭제하는 확률을 설정하는 것임.



드롭아웃을 구현하는 대표적인 방법 : 역 드롭아웃



테스트에서 드롭아웃을 구현하지 않음.

드롭아웃의 이해

직관 : 어떤 특정한 feature에 의존할 수 없다. 가중치를 다른 곳으로 분산시키게 됨.

→ 가중치의 노름 제곱값이 줄어듦.

- keep.prob을 정해야 함.

과대 적합의 위험성이 있는 레이어에 낮은 keep.prob 부여.

- input layer에도 드롭아웃 적용 가능

→ 하지 않는 것이 좋음. 입력 특성을 많이 없애고 싶지 않기 때문임.

- 과대 적합이 일어났을 때 드롭아웃 사용.

단점 : 비용함수 J가 더이상 잘 정의되지 않는다.

반복마다 드롭아웃을 하여 무작위로 한 뭉치의 노드들을 삭제하게 되고, 경사 하강법의 성능을 이중으로 확인한다면

→ 모든 반복에서 잘 정의된 비용함수가 하강하는지 확인하는 게 어려워짐.

→ 따라서 우선 드롭아웃을 사용하지 않고, 비용함수가 단조감소인지 확인 후에 사용해야 함.

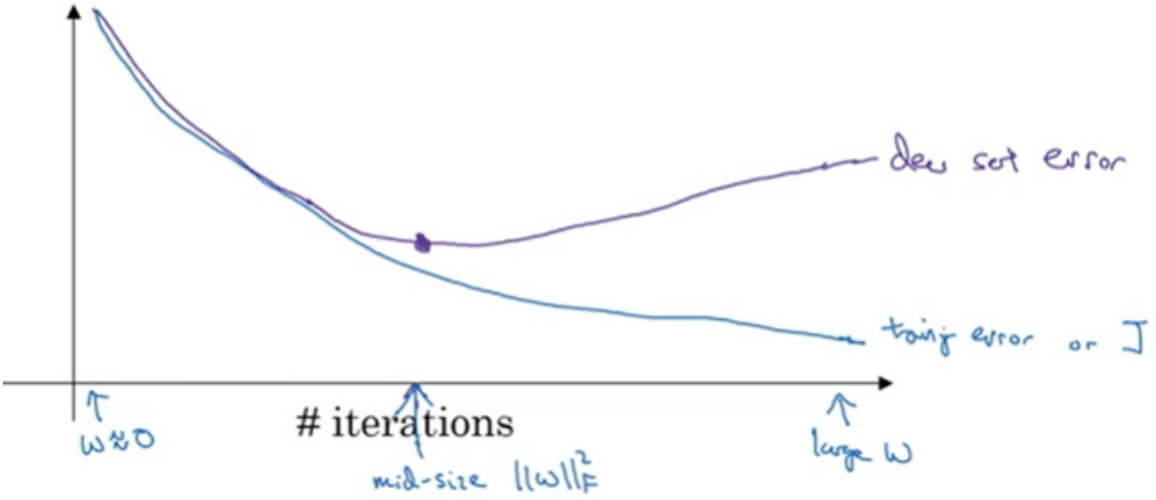
다른 정규화 방법들

- Data augmentation (데이터 증식)

사진을 뒤집거나 이미지를 편집(확대, 돌리기 등) 등

→ 비용이 들지 않음.

- Early stopping (조기 종료)



단점 : 훈련시 훈련 목적인 비용 함수를 최적화 시키는 작업과 과대적합하지 않게 만드는 작업이 있다. 두 작업은 서로 다른 방법으로 접근해야 한다. 그러나 조기 종료는 두 가지를 섞어 버리기 때문에, 최적의 조건을 찾지 못할 수도 있다.

- 경사 하강법을 일찍 멈춤으로써 비용함수를 최적화하는 것을 멈추게 함.
- 동시에 과대 적합하지 않으려고 함.

→ 대안 : L2 정규화

하지만 이 방법의 단점은 정규화 매개변수 람다에 많은 값을 시도해야 한다는 것임. → 컴퓨터적으로 비용이 많이 듦.