1.Train/Dev/Test Sets

훈련, 개발, 테스트 데이터 셋을 어떻게 설정할지에 관한 좋은 선택을 내리는 것은 좋은 성능을 내는 네트워크를 빠르게 찾는데 큰 영향을 준다.
신경망을 훈련시킬 때 결정해야 하는 것들은 다음과 같이 매우 많다.

- 층의 수(number of layers)
- 뉴런의 수(number of hidden units)
- 학습률(learning rate)
- 활성화 함수(Activation Function)

하지만 이 모든 것들에 대한 적절한 값을 처음부터 추측하는 것은 거의 불가능하다.
이 외에 다른 하이퍼파라미터(hyper parameters)도 마찬가지이다.
따라서 실질적으로 머신러닝을 적용하는 것은 매우 반복적인 과정을 거쳐야 한다.

1) 처음에는 아이디어로 시작한다.
2) 그리고 특정 개수의 층과 유닛을 가지고 특정 데이터 셋에 맞는 신경망을 만든다.
3) 이를 코드로 작성하고 실행하고 성능을 검사한다.
4) 그 결과에 기반하여 아이디어를 개선하고 몇 가지 선택을 수정하게 된다. 그리고 더 나은 신경망을 찾기 위해 이 과정을 반복한다.

신경망은 자연어 처리, 컴퓨터 비전, 음성 인식 등 여러 분야에 기여를 하였다. 하지만 어떤 분야에 적용되고 사용되는 직관이 다른 영역에도 적용되지는 않는다. 따라서 많은 분야에서 딥러닝에 아주 경험이 많은 사람일지라도 첫 시도에 하이퍼파라미터에 대한 최고의 선택을 올바르게 추측하는 것은 거의 불가능하다.

매우 반복적인 과정 속에서 빠른 진전을 이루는데 영향을 미치는 것들은 이 사이클을 얼마나 효율적으로 돌 수 있는지와 데이터 셋을 잘 설정하는 것이다. 훈련, 개발, 테스트 셋을 잘 설정하는 것은 과정을 더 효율적으로 만든다.

전통적인 방법으로는 모든 데이터를 가져와서 일부를 잘라서 훈련 셋으로 만들고 다른 일부는 교차 검증(또는 개발=dev) 셋으로 만들고 나머지 부분은 테스트 셋으로 만든다.

머신러닝 이전 시대의 관행적으로는 훈련 데이터 셋을 70%, 테스트 데이터 셋을 30%으로 한다. 혹은 훈련, 개발, 테스트 셋을 각각 60%, 20%, 20%로 나눈다. 데이터 샘플의 수가 100개, 1000개, 혹은 10000개였을 경우에는 합당한 비율이었다. 하지만 총 100만 개 이상의 샘플이 존재하는 현대 빅데이터 시대에는 개발 셋과 테스트 셋을 훨씬 더 작은 비율로 나누는 것이 트렌드가 되었다.

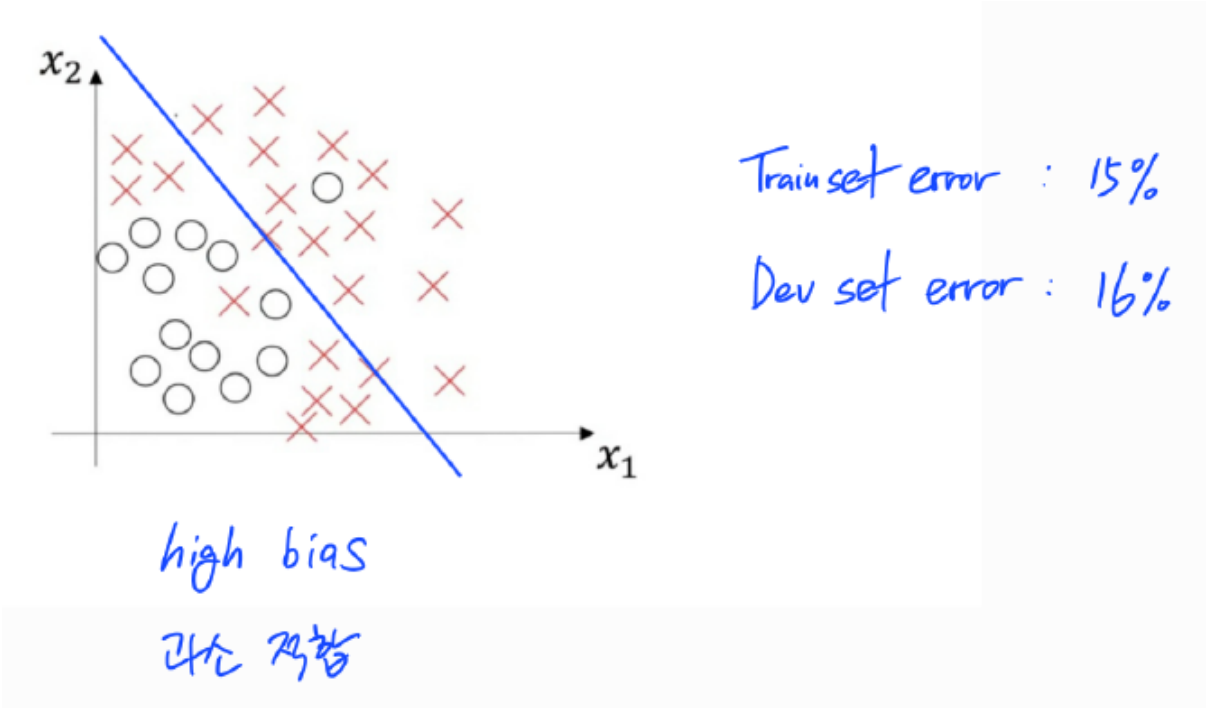
둘은 확인의 용도이므로 성능을 평가하는 정도로만 쓰면 되기 때문이다. 따라서 예를 들어 백만 개의 샘플이 있다면 개발과 테스트 데이터 샘플을 각각 만 개 정도로 설정해도 충분하다. 즉, 98%의 훈련 셋, 1% 개발 셋, 1% 테스트 셋가 된다. 데이터가 더 많다면 개발과 테스트 셋을 이보다 더 적게 설정해도 된다.

---

2. Bias/Variance

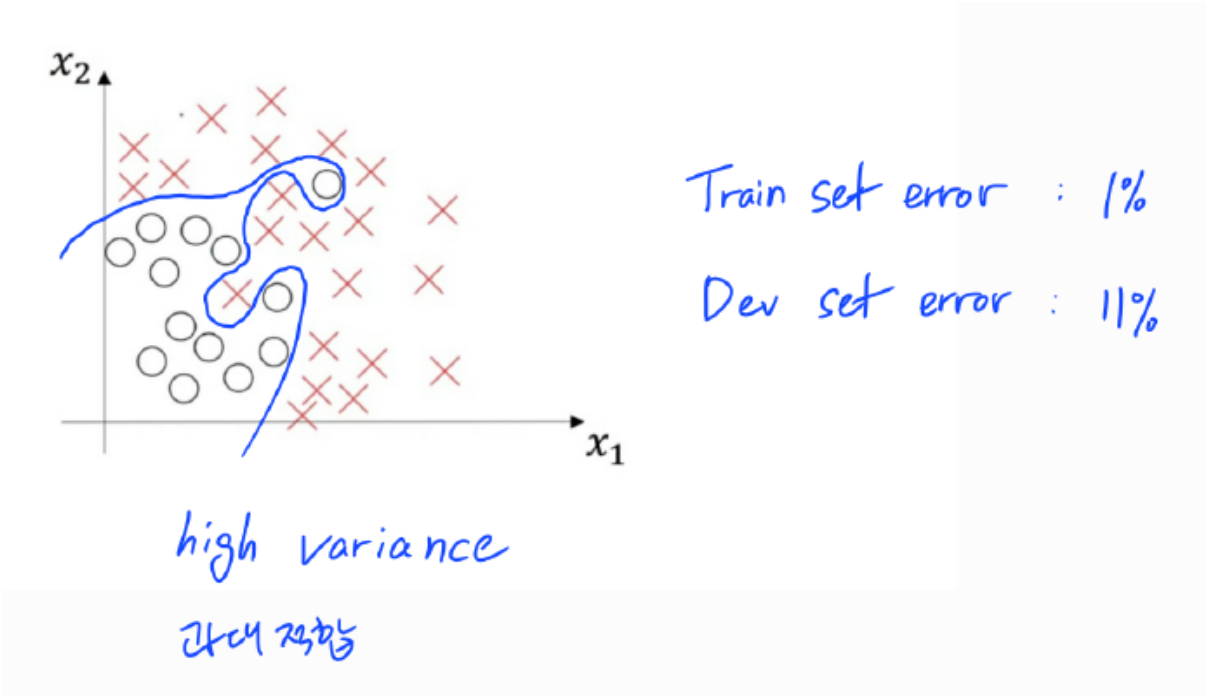
아래의 사례들을 통해 편향(Bias)과 분산(Variance)에 대한 개념을 완벽히 이해하려 한다.

1) 높은 편향(High Bias)



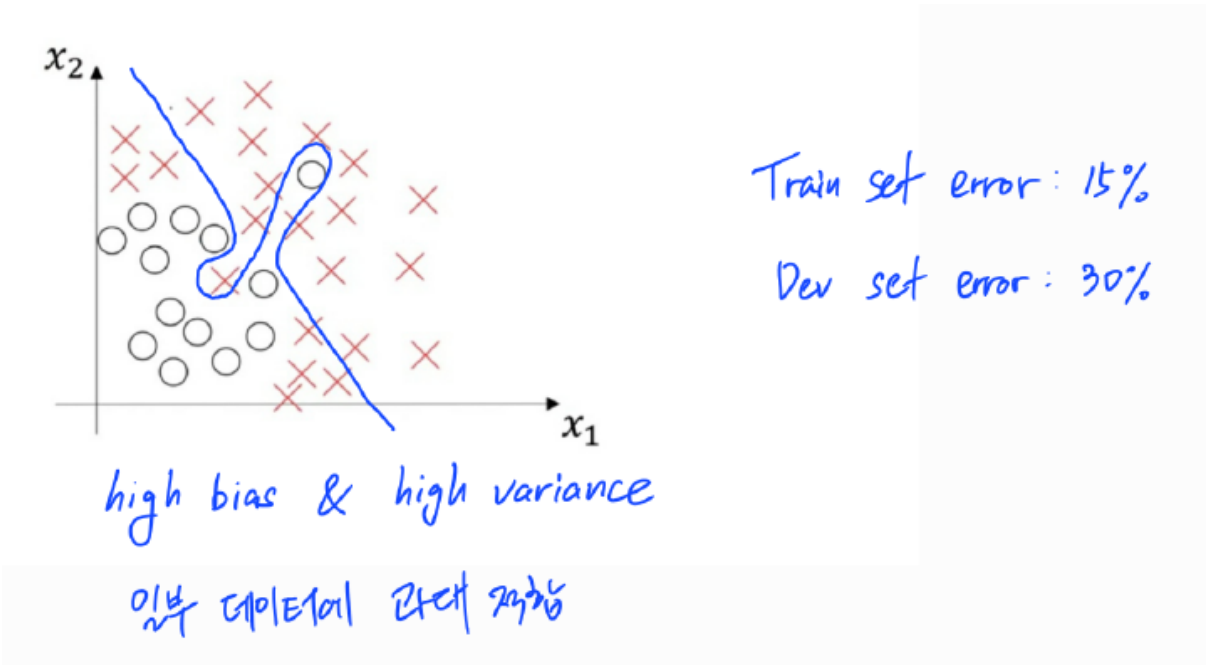
훈련 셋에 대하여 위와 같이 O와 X를 분류하였다면 이는 제대로 분류해내지 못한 과소적합(Underfitting)된 상황임을 알 수 있다. 훈련 셋 오차가 15%이고 개발 셋 오차가 16% 정도 나왔을 경우가 이에 해당한다.

2) 높은 분산(High Variance)



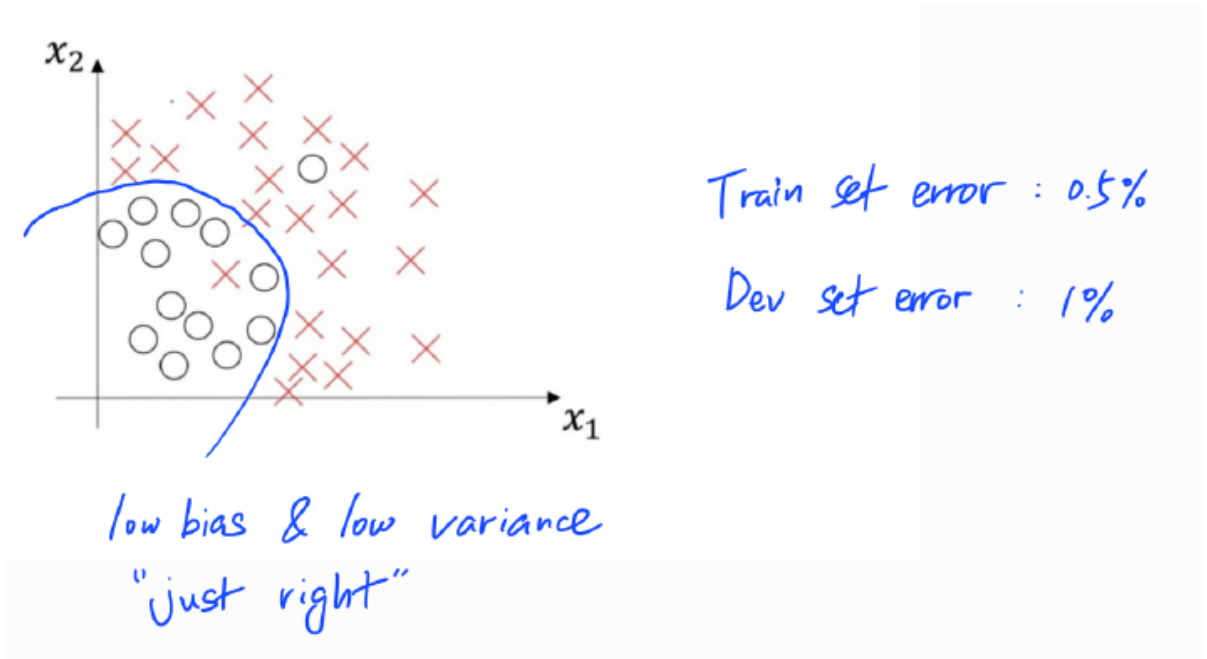
위의 사례의 경우에는 훈련 데이터 셋에 너무 과하게 최적화되고 일반화가 전혀 되지 않은 과대적합(Overfitting)한 상황이다. 훈련 셋 오차가 1%, 개발 셋 오차가 11% 정도 나왔다면 이러한 상황에 처한 것이다.

3) 높은 편향(High Bias) & 높은 분산(High Variance)



위의 경우에는 편향도 높고 분산도 높은 최악의 상황으로 볼 수 있다. 훈련 데이터 셋 전체에 최적화되지도 못했을 뿐더러 극히 일부 데이터에만 과대적합된 상황이다. 따라서 학습 셋 오차도 높고 개발 셋 오차는 더 높게 나올 것이다.

4) 낮은 편향(Low Bias) & 낮은 분산(Low Variance)



이 사례가 가장 훈련 셋에 최적화도 잘 되고 개발 셋에도 일반화가 잘 될 성능이 가장 좋은 경우이다. 적당히 적합(fit)한 모델이라고 볼 수 있다. 훈련 셋과 개발 셋 모두에서 오차가 매우 작게 나올 것이다.

단, 위의 분석은 인간 수준의 성능의 오차가 거의 0%라는 가정에 근거한다. 더 일반적으로는 최적의 오차, 가끔은 베이지안 오차라고 불리는 베이지안 최적 오차가 거의 0%라는 가정이다. 즉, 인간의 경우 예측의 정확도가 100%라는 것이다. 예를 들어 이미지를 분류할 때 이미지가 흐릿하여 인간 또한 잘 분류하지 못하여 최적 오차 또는 베이지안 오차가 15%인 경우에는 이야기가 달라진다. 이때는 훈련 셋의 오차 약 15%이고 개발 셋 오차가 16% 정도라면 매우 좋은 성능을 가지는 모델이다.

결론적으로 훈련 데이터 셋에서 개발 데이터 셋으로 갈 때 오차가 얼마나 커지는지에 따라서 분산과 편향 문제가 얼마나 심각한지에 대한 감을 잡을 수 있다. 다시 말해, 훈련 데이터 셋에서 개발 데이터 셋으로 일반화를 잘 하느냐에 따라 분산과 편향에 대한 감이 달라진다고 볼 수 있다. 단, 이는 베이지안 오차가 꽤 작고 훈련 셋과 개발 셋이 같은 확률 분포에서 왔다는 가정 아래에 이루어진다. 분산과 편향을 확인하여 현 상황을 진단하고 이후 적절한 조치를 취할 수 있어야 한다.

---

3. Basic Recipe for Machine Learning

신경망 훈련 기본 레시피

1. 편향(Bias) 확인

먼저 편향의 정도를 확인한다. 편향을 확인하기 위해서는 훈련(Train) 데이터 셋의 성능(오차)를 확인한다. 편향이 높다면 훈련 셋에도 최적화가 되지 않는다. 높은 편향을 보인다면 다음과 같은 방법을 이용하여 해결할 수 있다.

- (더 많은 은닉층 혹은 은닉 유닛을 갖는) 큰 네트워크 선택
- Train longer or 더 발전된 최적화 알고리즘 사용
- 다른 신경망 아키텍쳐 사용

위와 같은 방법들을 편향 문제를 해결할 때까지 반복한다. 보통 충분히 큰 네트워크라면 훈련 데이터에는 최적화시킬 수 있다. 물론 과대적합 될 수도 있다. 단, 이미지가 흐릿한 경우(베이지안 오차가 높다면)에는 애초에 최적화가 불가능하다.

2. 분산(Variance) 확인

편향 문제를 해결한 다음은 분산 문제가 있는지 확인해야 한다. 즉, 꽤 좋은 훈련 셋 성능에서 꽤 좋은 개발 셋 성능을 일반화할 수 있는지를 평가해야 한다. 이를 평가하기 위해서는 개발(Dev) 데이터 셋의 성능을 확인한다. 높은 분산을 가진다면 다음의 방법들을 시도해 볼 수 있다.

- 더 많은 데이터 확보
- Regularization(정규화)
- 다른 신경망 아키텍쳐 사용

이 방법들을 낮은 편향과 분산을 찾을 때까지 계속 반복하여 시도한다.

편향-분산 트레이드오프(Trade-off)

초기 머신러닝의 시대에는 편향-분산 트레이드오프에 대한 논의가 있었다. 시도할 수 있는 많은 방법들이 편향을 증가시키고 분산을 감소시키거나, 편향을 감소시키고 분산을 증가시키기 때문이다. 그러나 현대의 딥러닝 빅데이터 시대에는 더 큰 네트워크를 훈련시키는 방법(높은 편향 해결)과 더 많은 데이터를 얻는 방법(높은 분산 해결)이 항상 적용되지는 않지만 트레이드오프를 크게 발생시키지 않는다. 즉, 더 큰 네트워크를 사용하면 분산을 해치지 않고 편향만이 감소하며, 더 많은 데이터를 얻는 것도 편향을 해지치 않고 분산만을 감소시킨다.

Regularization(정규화)는 또다른 분산을 줄이는 유용한 기술이다. 정규화를 사용하면 편향을 조금 증가시킬 수 있어 약간의 편향-분산 트레이드오프가 존재한다. 하지만 충분히 큰 네트워크를 사용한다면 그렇게 크게 증가하지는 않는다.

결론적으로, 신경망을 훈련시킬 때 먼저 편향을 확인하고 문제가 있을 시 이를 여러 방법론으로

해결한 뒤, 분산을 확인하고 또 문제가 있을 시 여러 방법들을 시도하여 해결한다. 각각을 해결하기 위한 적절한 방법들은 서로 다를 수 있으니 정확히 상황을 파악하고 적합한 해결책을 사용해야 효율적으로 신경망을 구현할 수 있다.
또한 초기에는 편향-분산 트레이드오프가 매우 중요한 이슈였으나 딥러닝이 매우 발달한 현대에는 편향과 분산이 서로 영향을 미치지는 않는다.