6주차

날짜	@2024년 4월 16일
과제	강의 요약 출석 퀴즈
세부내용	[딥러닝 2단계] 1. 머신러닝 어플리케이션 설정하기 2. 신경망 네트워크의 정규화
자료	[Week6] 출석퀴즈 C2_W1.pdf

머신러닝 어플리케이션 설정하기

1️⃣ Train/Dev/Test 세트

Applied ML is a highly iterative process

- 신경망 훈련 시 결정해야 하는 것
  - 신경망의 층 수
  - 각 층의 hidden units 수
  - 학습률
  - 층마다의 활성화 함수 종류
- 좋은 하이퍼파라미터 값을 찾기 위해 사이클 반복

  → 아이디어 → 코드 작성/실행 → 실험 진행 → 아이디어 개선/선택 변경
- 빠르게 진행하는 데에 영향을 미치는 것
  - 사이클을 얼마나 효율적으로 돌 수 있는지
  - 데이터셋을 얼마나 잘 설정했는지

Train/Dev/Test sets

- 전통적인 방법
  - 전체 데이터를 training set / Hold-out 교차 검증, Dev set / test set으로 분리
  - 훈련 세트에서 훈련 알고리즘 적용
  - 개발 세트(교차 검증 세트)에서 가장 좋은 성능을 나타내는 모델을 확인
  - 테스트 세트에 최종 모델을 적용
- ~ 머신러닝 시대 : 70(train)/30(test) or 60(train)/20(dev)/20(test)로 분할
- Big data 시대 : 100만 개 이상의 샘플 존재
  - dev/test set의 비율을 더 작게 설정하 것이 좋음
  - 100만 개 : 98(train)/1(dev)/1(test)
  - 100만 개 이상 : 99.5%/0.4%/0.1%

Mismatched train/test distribution

- 일치하지 않는 train/test distribution에서 훈련
- training set : cat pictures from webpages → 상대적으로 잘 정돈된 사진
- dev/test set : cat pictures from users → 상대적으로 저해상도 사진

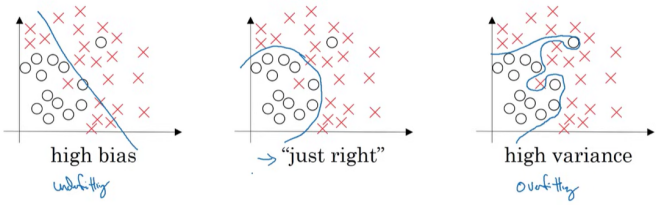
- 두 데이터는 분포가 다를 수 있음

  ⇒ dev와 test set은 같은 분포에서 와야 함
- 테스트 세트를 가지지 않아도 괜찮음 (비편향 추정이 필요 없다면?)

  → train에서 훈련 dev에서 성능 평가하여 가장 좋은 성능의 모델을 찾음

2️⃣ 편향/분산

Bias and Variance



- 높은 편향 ↔ 과소적합
- 높은 분산 ↔ 과대적합

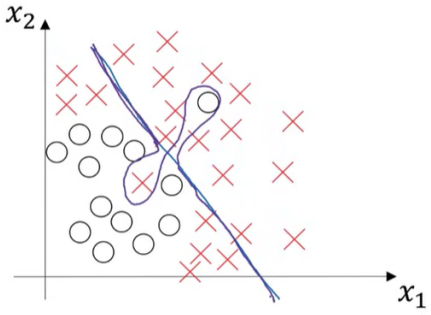
Cat Classification 예제

- y=1 (고양이), y=0 (고양이 x)
- 베이지안 최적 오차 = 0% 가정 & 훈련셋과 개발셋이 같은 분포에서 옴

train set error	dev set error	결과
1%	11%	과대적합 → 높은 분산
15%	16%	과소적합 → 높은 편향
15%	30%	높은 편향 & 높은 분산
0.5%	1%	낮은 편향 & 낮은 분산

- 훈련 세트 오차를 통해 훈련 데이터에 얼마나 알고리즘이 적합한지 확인 → 편향 문제 점검
- 훈련 세트에서 개발 세트로 갈 때 오차가 얼마나 커지는지 확인 → 분산 문제 점검

High bias and high variance



- 일부 데이터에 대해서만 과대적합 하는 경우

  → high bias, high variance

3️⃣ 머신러닝을 위한 기본 레시피

Basic recipe for machine learning

- 높은 편향을 가지는지? (훈련 데이터 성능 확인)
  - 더 많은 은닉층 or 은닉 유닛을 갖는 네트워크 선택
  - 더 오랜시간 훈련
  - 다른 최적화 알고리즘 선택
  - (더 적합한 신경망 아키텍처 선택)
- 높은 분산을 가지는지? (개발 데이터 성능 확인)
  - 데이터를 더 얻는 것
  - 과대적합을 줄이기 위한 정규화 시도
  - (더 적합한 신경망 아키텍처 선택)

⇒ 편향 or 분산 or 편향과 분산 모두 문제가 있는지 파악하는 것이 중요

- 딥러닝 시대에서 "편향과 분산의 trade off"
  - 더 큰 네트워크를 갖는 것 : 분산을 해치지 않고 편향을 감소
  - 데이터를 더 얻는 것 : 편향을 해치지 않고 분산을 감소

    ⇒ trade off 현상이 적은 방법

신경망 네트워크의 정규화

1️⃣ 정규화

Logistic Regression

- 로지스틱 회귀 : 비용함수 J를 최소화하는 매개변수를 찾는 것
  - $J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$
- 로지스틱 회귀에 정규화를 추가
  - $J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}||w||_2^2$
- L2 정규화 → w에 대해 유클리드(L2) norm을 사용
  - $||w||_2^2 = \sum_{j=1}^{n_x} w_j^2 = w^T w$
- L1 정규화
  - $\frac{\lambda}{2m}||w||_1 = \frac{\lambda}{2m}\sum_{i=1}^{n_x} |w|$
  - w가 희소해짐(=w 벡터 안에 0이 많아짐) → 모델 압축에 유용

  ⇒ L1보다는 L2를 주로 사용
- b에는 norm을 적용하지 않는 이유?
  - w는 보통 높은 차원의 매개변수 벡터 (특히 높은 분산을 가질 때)
  - b는 하나의 숫자
  - 거의 모든 매개변수는 w에 존재하기 때문에 b에 norm을 적용하는 것은 큰 영향을 미치지 않음
- λ : 정규화 매개변수(하이퍼파라미터)
- 두 매개변수의 노름을 잘 설정하여 과대적합을 막을 수 있는 최적의 값을 찾음

Neural Network

-

- $J(w^{[1]}, b^{[1]}, \cdots, w^{[L]}, b^{[L]}) = \frac{1}{m}\sum_{i=1}^{n} L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L} ||w^{[l]}||^2$
- Frobenius norm : $||w^{[l]}||_F^2 = \sum_i \sum_j (w_{ij}^{[l]})^2$, where w : $(n^{[l-1]}, n^{[l]})$

  → 행렬의 L2 노름이라고 부르지 않고 Frobenius norm이라고 부름
- 정규화항을 더해준 후 역전파
  - $dw^{[l]} = (\text{from backpropagation}) + \frac{\lambda}{m} w^{[l]} = \frac{\partial J}{\partial w^{[l]}}$
  - 기존 비용함수 미분에 정규화 항만 더해준 것
- L2 정규화과 가중치 감쇠라고 불리는 이유
  - $w^{[l]} = w^{[l]} - \alpha(\text{from backpropa} + \frac{\lambda}{m} w^{[l]}) = (1 - \frac{\alpha\lambda}{m}) w^{[l]} - \alpha(\text{from backpropa})$
  - $w^{[l]}$ 값보다 더 작아지게 됨

2️⃣ 왜 정규화는 과대적합을 줄일 수 있을까요?

How does regularization prevent overfitting?

- 과대적합 문제가 있는 신경망
  - $J(w^{[l]}, b^{[l]}) = \frac{1}{m}\sum_{i=1}^{n} L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L} ||w^{[l]}||_F^2$
  - λ를 크게 하면 가중치 행렬 w를 0에 가깝게 설정할 수 있음
  - 많은 은닉 유닛을 0에 가까운 값으로 설정해서 은닉 유닛의 영향을 줄이면 간단한 신경망이 됨

    ⇒ 분산의 감소 (과적합 방지)
- g(z) = tanh(z)인 경우
  - z가 아주 작은 경우 tanh 함수의 선형 영역을 사용
  - z의 값이 커지거나 작아지면 선형을 벗어남
  - λ가 커질 때 비용함수가 커지지 않으려면 상대적으로 w가 작아짐

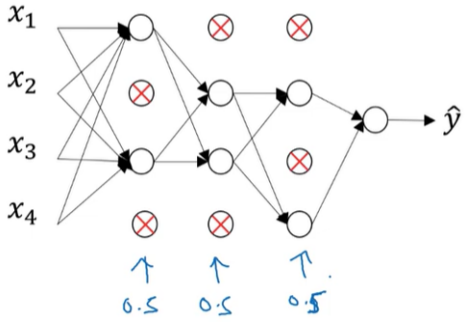
    → z도 상대적으로 작은 값을 갖게 됨 → g(z)는 거의 1차원 함수

    ⇒ 모든 층이 선형 회귀와 같은 선형 함수를 갖게 됨 → 과적합 방지

  ⇒ λ ↑ → w ↓ → z ↓ → 선형 함수 → 과적합 가능성 감소
- 경사하강법 구현 시 정규화 매개변수가 포함되었는지 확인 !

3️⃣ 드롭아웃 정규화

Dropout regularization



- 드롭아웃 방식 : 신경망 각 층에 대해 노드를 삭제하는 확률을 설정
  - 각 층에 대해 각 노드마다 0.5 확률로 유지할지 삭제할지 결정

- 삭제된 노드의 들어가는 링크와 나가는 링크를 모두 삭제
- 감소된 네트워크에서 하나의 샘플을 역전파로 훈련
- 각 훈련 세트에 대해 드롭아웃 적용

⇒ 각 샘플에서 더 작은 네트워크를 훈련시키는 방식

Implementing dropout ("Inverted dropout")

- layer = 3, keep_prob = 0.8(주어진 은닉 유닛이 유지될 확률)
- d3 = np.random.rand(a3.shape[0], a3.shape[1]) < keep_prob

  → 0.8의 확률로 대응하는 d3가 1의 값을 가지고, 0.2의 확률로 대응하는 d3가 0의 값을 가지는 행렬 생성

  ⇒ d3 : True / False를 갖는 bool type 행렬
- a3 = np.multiply(a3,d3) # a3 *= d3

  → 모든 원소에 대해 20% 확률로 0이 되는 d3의 원소를 곱해 대응되는 a3 원소를 0으로 만듦
- a3 /= keep_prob ⇒ 역드롭아웃
  - 세 번째 은닉층에 50개의 유닛이 있다고 가정
  - a3 (50, 1) -vectorization→ (50, 3) 차원
  - 80% 유지, 20% 삭제 → 평균적으로 10개의 유닛 삭제, 즉 0 값을 가짐
  - $z^{[4]} = w^{[4]} * a^{[3]} + b^{[4]}$

    → $a^{[3]}$의 원소의 20%가 0이 됨

    → $z^{[4]}$의 기댓값에 영향을 주지 않기 위해서 $a^{[3]}$을 0.8로 나누어줘야 함
  - keep_prob을 다시 나눠줌으로써 a3의 기댓값을 같게 함
- 스케일링 문제가 적어서 테스트를 쉽게 해 줌
- 가장 보편적인 드롭아웃 방법
- 계속 같은 유닛을 0으로 하는 것이 아니라 경사 하강법의 한 반복마다 0이 되는 은닉 유닛이 달라짐

Making predictions at test time

- $a^{[0]} = X$
- test에서는 drop-out을 사용하지 않음
  - 예측을 해야하기 때문에 결과가 무작위로 나오는 것을 원하지 않음
  - test에서의 drop-out 구현은 노이즈만 증가시킴
- 역드롭아웃 : 테스트에서 드롭아웃을 수행하지 않아도 활성화 기대값의 크기가 변하지 않기 때문에 테스트 시 스케일링 매개변수를 추가하지 않아도 됨

4️⃣ 드롭아웃의 이해

Why does drop-out work?

- 랜덤으로 노드를 삭제해서 하나의 특성에 의존하지 못하도록 함
- 가중치를 분산시켜 가중치의 노름의 제곱값이 줄어들게 됨
- 드롭아웃 효과 : 가중치를 감소해서 정규화하는 것 → L2 정규화와 비슷한 효과
- keep_prob
  - 층마다 달리할 수 있음

    - 가장 많은 매개변수 = 가장 큰 가중치 행렬 → 상대적으로 낮은 keep_prob 설정, 강력한 드롭아웃
    - 과대적합의 우려가 적은 층 → 더 높은 keep_prob 설정 가능
    - 과대적합 우려가 없는 층은 1로 설정 → 모든 유닛 유지, 드롭아웃 사용하지 않는다는 의미
- 입력층에서도 적용 가능 → 보통 1에 가깝게
- 컴퓨터 비전 : 데이터 부족 → 기본값으로 드롭아웃 사용

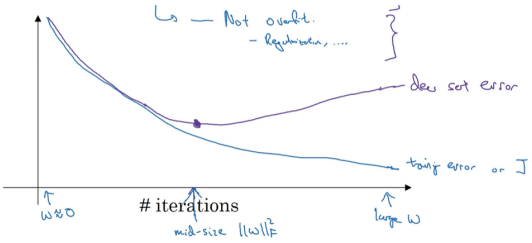
  ⇒ 드롭아웃은 정규화 기법, 과대적합이 발생했을 때 사용
- 단점 : 경사하강법에서 비용함수 J가 잘 정의되지 않음 → 디버깅 어려움
    - 드롭아웃 없이 먼저 J가 단조감소하는지 확인한 후 드롭아웃 사용

5️⃣ 다른 정규화 방법들

Data augmentation (데이터증식)

- 이미지 : 더 많은 훈련 데이터를 사용해서 과적합 해결
- 이미지를 수평 방향으로 뒤집기, 확대, 회전, 왜곡 등으로 새로운 데이터 추가
- 새 데이터를 찾는 것보다는 적은 정보 but 많은 비용이 들지 않음
- 뒤집어진 고양이도 고양이다 !
- 구부러진 숫자도 숫자다 !

Early stopping



- 경사하강법 실행 →훈련 오차/비용함수 단조 감소
- 개발 세트 오차도 함께 그려줌
- 개발세트 오차가 증가하기 시작하는 부분 → 과대적합 시작 지점
- 조기 종료를 통해 신경망이 개발 세트의 오차 저점 부분일 때 훈련을 멈춤