DL 스터디&프로젝트

[Euron 중급 세션 6주차] 딥러닝 2단계 2. 신경망 네트워크의 정규화

by 공부하자_ 2023. 10. 30.

딥러닝 2단계: 심층 신경망 성능 향상시키기

2. 신경망 네트워크의 정규화

📌정규화(C2W1L04)

핵심어: 정규화(regularization), 정규화 매개변수 \lamda, L1 norm, L2 norm, Frobenius norm

높은 분산으로 신경망이 데이터를 과대적합(오버피팅)하는 문제가 의심된다면 가장 처음 시도해야 할 것은 정규화이다. 높은 분산을 해결하는 다른 방법은 더 많은 훈련 데이터를 얻는 것인데, 이는 신뢰할만한 방법이지만 더 많은 훈련 데이터를 얻는 것은 비용이 많이 들어간다. 한편 정규화를 추가하는 것은 과대적합을 막고 신경망의 분산을 줄이는 데 도움이 많이 될 것이다. 정규화가 어떻게 작동하는지 살펴보도록 하자.

분류 전체보기

DL 스터디&프로젝트

Data Science 프로젝트

Github 스터디

Data Science 개인 공부

Backend 프로젝트

기타 공부

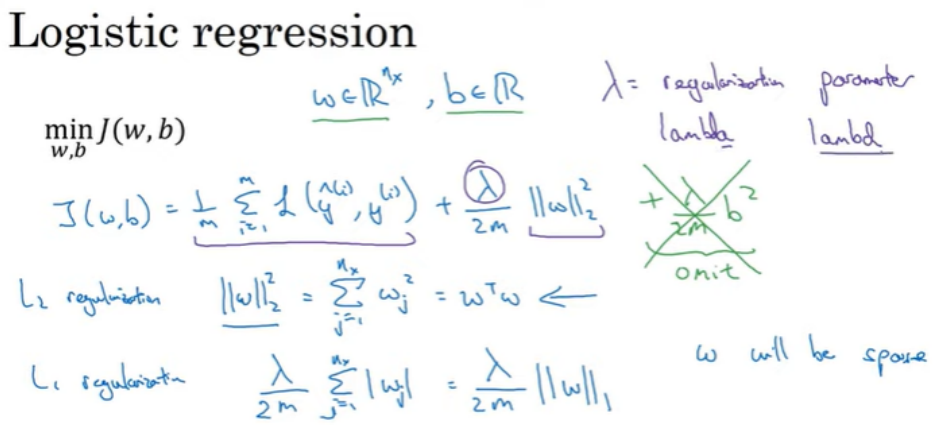
공지사항

최근글 인기글

[Euron 중급 세션 6주차] ... 2023.10.30

자프실 - OOP 개념 정리 2023.10.12

[Euron 중급 세션 5주차] ... 2023.10.09



로지스틱 회귀를 사용해 위의 아이디어를 발전시켜 보자. 로지스틱 회귀는 정의된 비용 함수 J를 최소화하는 것임을 기억해야 한다. 비용 함수 J(w, b)는 훈련 샘플의 개별적인 예측의 손실에 관한 함수이며, 로지스틱 회귀의 w와 b는 매개변수이다. w는 x차원의 매개변수이고 b는 실수이다. 따라서 로지스틱 회귀에 정규화를 추가하기 위해, 정규화 매개변수라고 부르는 람다 λ를 추가해야 한다. λ를 2m으로 나누고 w^2의 norm을 해준다. 여기서 w^2의 norm은 j의 1부터 nx까지 wj^2의 값을 더한 것과 같다. w의 전치행렬 곱하기 w와도 같다. 매개변수 벡터 w의 유클리드 norm의 제곱이 되는 것이다. 그리고 이것을 L2 정규하라고 부른다. 여기서 매개변수 벡터 w의 유클리드 norm, 즉 L2 norm을 사용하고 있기 때문이다.

왜 매개변수 w만 정규화하고 b에 관한 것은 추가하지 않는 것일까? 실제로 가능하지만 보통 생략한다. 보통 매개변수 w는 꽤 높은 차원의 매개변수 벡터이기 때문이다. 특히 높은 분산을 가질 때 w는 많은 매개변수를 갖는다. 반면에 b는 하나의 숫자이다. 따라서 거의 모든 매개변수는 b가 아닌 w에 있다. 이 마지막 항을 넣어도 실질적인 차이는 없을 것인데, 이는 많은 매개변수들 중 b는 하나의 매개변수일 뿐이기 때문이다.

L2 정규화는 가장 일반적인 정규화이다. L1 정규화에 대해서도 들어보았을텐데, L2 norm 대신 다음과 같은 항을 추가하는 것이다. λ를 m으로 나눈 값에 절댓값w의 합(시그마)을 더해준다. 이는 또한 매개변수 벡터 w의 L1 norm이라고 부른다(아래첨자 1). m 앞에 곱하는 2는 스케일링 상수이다. L1 정규화를 사용하면 w는 희소해지는데, 이는 w 벡터 안에 0이 많아진다는 의미이다. 누군가는 이것이 모델을 압축하는데 도움이 된다고 말하는데 이는 특정 매개변수가 0일 경우 메모리가 적게 필요하기 때문이다. 그러나 모델을 희소하게 만들기 위해 L1 정규화를 사용하는 것은 큰 도움이 되지 않는다. 따라서 모델을 압축하겠다는 목표가 있지 않는 이상 L1 정규화는 많이 사용하지 않는다. 사람들이 네트워크를 훈련할 때는 L2 정규화를 훨씬 더 많이 사용한다.

마지막으로 λ는 정규화 매개변수라고 부르는데, 개발 세트 혹은 교차 검증 세트를 주로 사용한다. 다양한 값을 시도해서 훈련 세트에 잘 맞으며 매개변수의 두 norm을 잘 설정해 과대적합을 막을 수 있는 최적의 값을 찾는다. 따라서 λ는 설정이 필요한 또 다른 하이퍼 파라미터이다(프로그래밍 예제를 할 때 lanmbda는 파이썬의 명령

백엔드 프로젝트 5주차 (SQL 첫걸음) - 5장
2023.10.07

백엔드 프로젝트 4주차 (SQL 첫걸음) - 4장
2023.10.07

최근댓글

좋은 글 잘 보고 가요! 감사...

좋은 글 잘 보고 가요! 감사...

잘보고 갑니다!

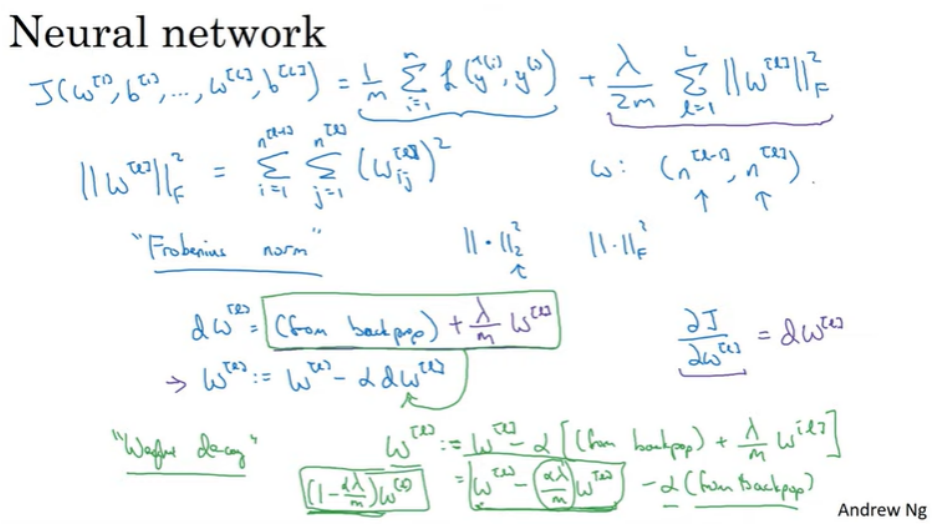
포스팅 잘보고 갑니다! 응원...

태그

딥러닝교과서, pandas, 데이터사이언스, 이지스퍼블리싱, 판다스, bda, 판다스입문, 딥러닝스터디, 데이터분석, Doit

전체 방문자
498
Today : 0
Yesterday : 1

어이며, mbd로 람다 정규화 매개변수를 나타낸다). 따라서 이것이 로지스틱 회귀에 대한 L2 정규화를 나타내는 방법이다.



신경망의 경우를 살펴보자. 신경망에도 비용 함수가 존재한다. 모든 파라미터 w[1], b[1]부터 w[L], b[L]까지의 매개변수를 갖는 함수이다. L은 신경망에 있는 층의 개수이다. 따라서 비용함수는 훈련 샘플의 m까지의 손실을 m으로 나눈 값이다. 여기에 정규화를 더하기 위해 λ를 2m으로 나눈 값 곱하기 매개변수 w norm 제곱의 모든 값을 더해준다. 행렬의 norm^2은 i와 j에 해당하는 각각의 행렬의 원소를 제곱한 것을 모두 더해준 값이다. 시그마의 범위를 정의하자면 i는 1부터 n[l-1]까지이고 j는 1부터 n[l]까지이다. 왜냐하면 w는 (n[l-1], n[l]) 차원의 행렬이기 때문이다. 이것은 해당 층 l-1과 l의 은닉 유닛의 개수를 나타낸다. 따라서 이 행렬 norm은 Frobenius norm이라고 부른다. 아래 첨자에 F 표시를 해준다. 선형대수학에서는 행렬의 L2 norm이라고 부르는 대신 Frobenius norm이라고 부르며 이는 행렬의 원소 제곱의 합이라는 뜻이다.

이를 경사하강법으로 구현하자면, 역전파를 생각하면 된다. 역전파 dw를 계산할 때, 역전파는주어진 l에 대한 w에 대응하는 J의 편미분 값을 제공했다. 그리고 w[l]을 w[l]- αdw[l]로 업데이터 하였다. 이것은 추가적인 정규화 하을 더해주기 이전의 값이다. 따라서 정규화 항을 더해주게 되면 dw[l]에 (λ/m)w[l]을 더해준다. 그리고 이 값을 전과 같은 방식으로 계산한다. 이 새로운 dw[l]의 정의는 여전히 비용함수의 미분에 대한 올바른 정의이다. 매개변수에 관해 끝에 정규화 항을 더해주는 것 뿐이다. 이러한 이유로 L2 정규화는 가중치 감쇠라고 불리기도 한다. 이 dw의 정의를 w[l] 업데이트 식에 적용시키면 w[l]은 w[l]- α((역전파에서 온 것들) + (λ/m)w[l]) 이 된다. 이 값은 <w[l]- (α*λ/m)w[l]- α(역전파에서 얻은 것들)>이 된다. 이 항은 행렬 w[l]이 어떤 값이든 값이 약간 더 작아진다는 것을 보여준다. 이는 w[l] 행렬의 값에 (1- (α*λ/m))을 곱해준 것과 같다. 다시 말해 행렬 w[l]에서 (α*λ/m)w[l] 만큼을 빼준 것이며, 행렬 w[l]의 값보다 더 작아지게 된다. 따라서 이것이 L2 norm 정규화가 가중치 감쇠라고 불리는 이유이다. w[l]에 α*dw[l]의 값을 빼서 값을 업데이트 하는 것은 원래의 경사 하강법과 동일하지만, 여기서는 또한 w[l]에 1보다 작은 값을 곱

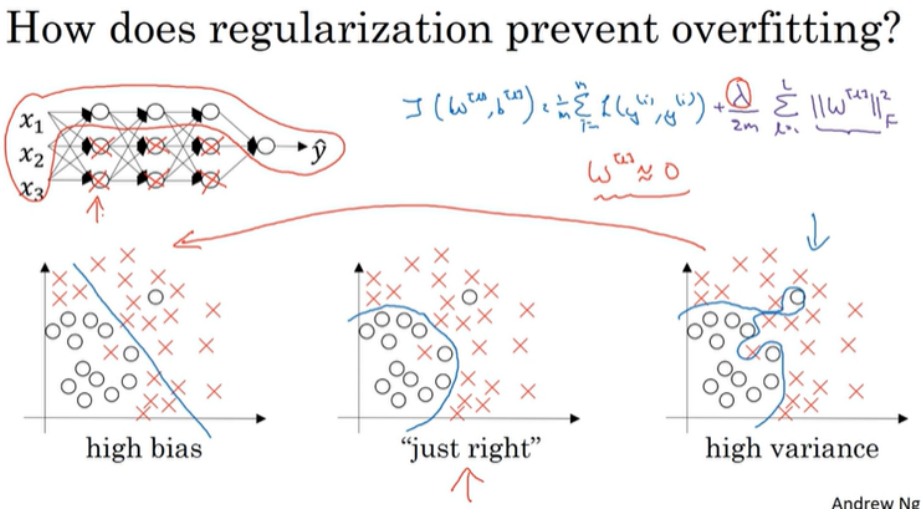
해주게 되는 것이며 이에 따라 L2 정규화의 또 다른 이름이 가중치 감쇠가 된 것이다. 이런 이름을 얻게 된 이유가 첫 번째 항이 (1- (α*λ/m))w[l] 값이기 때문이다. 즉 1보다 살짝 작은 값을 가중치 행렬에 곱해준다는 이유에서 나온 이름인 것이다.

- 비용함수: $J(w,b) = \frac{1}{m}\sigma_{i=1}^{m} L(\hat{y^{(i)}}, h^{(i)}) + \frac{\lambda}{2m}||w||^2$
- L1 정규화: $||w||_1 = \sigma_{j=1}^{nx}|w_j|$
- L2 정규화: $||w||_2^2 = \sigma_{j=1}^{nx} w_j^2$
- Frobenius 노름: $||w^{[l]}||_F^2 = \sigma_{i=1}^{n^{[l-1]}} \sigma_{j=1}^{n^{[l]}} (w_{ij}^{[l]})^2$
- L2 정규화가 weight decay 라고 불리는 이유는 아래와 같습니다.
  - $w^{[l]} = (1 - \frac{\alpha\lambda}{m}) w^{[l]} - \alpha$ (역전파에서 온 값들)
  - 위의 식을 보면 weight 에 1보다 작은 값인 ($1 - \frac{\alpha\lambda}{m}$) 가 곱해지기 때문입니다.

📌 왜 정규화는 과대적합을 줄일 수 있을까요?(C2W1L05)

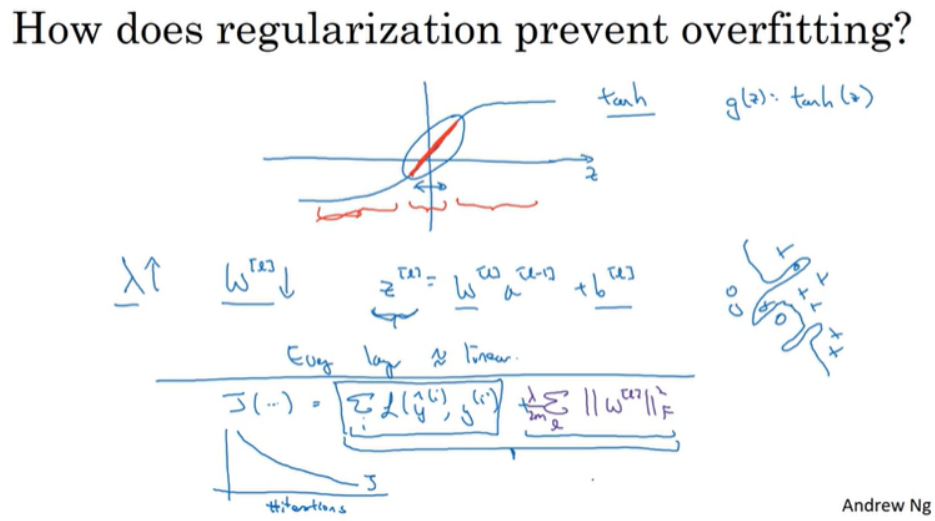
핵심어: 정규화(regularization), 과대적합(overfitting)

왜 정규화가 과대적합 문제를 해결하고 분산을 줄이는 데 도움이 될까? 어떻게 작동하는지에 관한 직관을 얻기 위해 예제를 살펴보자.



이전 강의의 높은 편향, 높은 분산 그리고 딱 맞는 경우의 예시들을 다뤘었다. 이번에는 크고 깊은 신경망에 맞추는 경우를 살펴보고자 한다. 과대적합의 문제가 있는 신경망을 생각해보자. w와 b에 대한 비용함수 J는 1부터 m까지 손실의 합과 같다. 정규화를 위해 추가적인 항 λ/2m* Σ ||w||^2을 더해주었다. 가중치 행렬이 너무 커지지 않도록 막기 위해서이다. 그리고 이것을 Frobenius norm이라고 부른다. 그렇다면 매개변수의 L2 혹은 Frobenius norm을 줄이는 것이 왜 과대적합을 줄일 수 있을까? 여기서 알 수 있는 것은 정규화에서 λ를 크게 만들어서 가중치 행렬 w를 0에

상당히 가깝게 설정할 수 있다는 것이다. 따라서 많은 은닉 유닛을 0에 가까운 값으로 설정해서 은닉 유닛의 영향력을 줄인다. 그런 경우 훨씬 더 간단하고 작은 신경망이 될 것이다. 이때 다양한 층을 저장하고 있긴 하지만, 로지스틱 회귀 유닛에 가까워진다. 따라서 이런 과적합한 경우를 왼쪽의 높은 편향의 경우와 가깝게 만들어 줄 수 있다. 그러나 중간의 딱 맞는 경우와 가깝게 하는 적절한 λ값을 찾는 것이 좋다. λ의 값을 아주 크게하면 w는 0에 가깝게 설정된다는 것이다. 실제의 경우에서는 은닉 유닛의 영향력을 0에 가깝게 줄임으로써 로지스틱 회귀에 가까운 네트워크를 만든다고 생각하면 될 것이다. 한 뭉치의 은닉 유닛을 완전히 0으로 만든다는 것은 아니며, 실제로 모든 은닉 유닛을 사용하지만 각각의 영향력이 훨씬 작아진 것이다. 더 간단한 네트워크가 되는 것은 맞으며 간단한 네트워크는 과대적합 문제가 덜 일어나게 된다. 프로그래밍 예제에서 정규화를 구현할 때 이와 같은 분산의 감소를 결과를 발견하게 될 것이다.



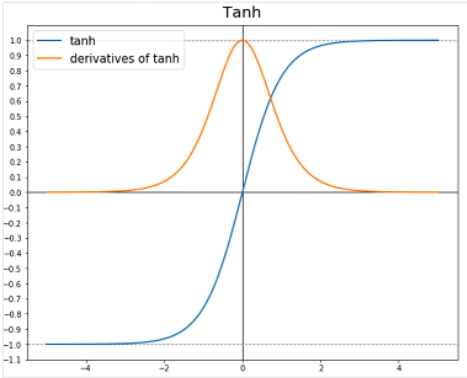
왜 정규화가 과대적합을 막는 데 도움을 주는지에 대한 또 다른 직관이 있다. tanh 활성화 함수를 사용한다고 가정할 때, 즉 g(z)=tanh(z)일 때 z가 아주 작은 경우, z가 작은 범위의 매개변수를 갖는 경우라면 tanh 함수의 선형 영역을 사용하게 된다. z의 값이 더 작아지거나 커지면 활성화 함수는 선형을 벗어나게 된다. 여기서 얻을 수 있는 직관은, 정규화 매개변수인 λ가 커지면, 비용 함수가 커지지 않으려면 상대적으로 w가 작아질 것이며 가중치 w[l]이 작으면 z[l]이 w[l]a[l-1]+b[l]과 같기 때문에 w가 작으면 z도 상대적으로 작은 값을 가지게 되는 것이다. 이 작은 범위에서 z가 상대적으로 작은 값을 가지게 된다면 g(z)는 거의 1차함수(linear)가 될 것이다. 따라서 모든 층은 선형 회귀처럼 거의 직선의 함수를 갖게 된다. 모든 층이 선형이면 전체 네트워크도 선형이 되는데, 이때 선형 활성화 함수를 가진 깊은 네트워크의 경우에도 선형 함수만을 계산할 수 있게 된다. 따라서 매우 복잡한 결정, 비선형 결정의 경계에 맞추기는 불가능하다. 앞 슬라이드의 높은 분산의 경우처럼 과대적합된 데이터 세트까지 맞추기는 어려운 것이다. 정리하자면 정규화 매개변수가 매우 크면 매개변수 w는 매우 작고, b의 효과를 무시하면 z의 값은 상대적으로 작다. z가 상대적으로 작은 범위의 값을 가지기 때문에 tanh의 경우 활성화 함수는 상대적으로 선형이 되며 전체

신경망은 선형 함수로부터 그리 멀지 않은 곳에서 계산될 것이다. 즉 매우 복잡한 비선형 함수보다 더 간단한 함수인 것이며, 이에 따라 과대적합의 가능성이 줄어든다.

정규화에 관한 논쟁을 마무리하기 전에 구현에 관한 팁을 하나 드리고자 한다. 정규화를 구현할 때 비용함수 J의 정의를 살펴보았는데, 여기에 가중치가 너무 커지는 것을 막기 위한 추가적인 항을 추가하였다. 경사 하강법을 구현할 때, 경사 하강법을 정의하는 한 가지 단계는 경사 하강법의 반복의 수에 대한 함수로 비용함수를 설정하는 것이며 이에 비용함수 J가 경사 하강법의 반복마다 단조감소하기를 원할 것이다. 정규화를 구현할 때 비용함수 J의 예전 정의, 즉 첫 항만을 그린다면 단조 감소를 보지 못할 것이다. 따라서 경사하강법을 디버깅할 때는 두 번째 항을 포함하는 새로운 비용함수 J를 그리고 있는지 잘 확인하는 것이 좋다. 만약 그렇지 않다면 매 반복에서 J가 단조감소하는 것을 보지 못할 것이다.

- λ 값을 크게 만들어서 가중치행렬 w 를 0에 가깝게 설정할 수 있습니다.
  - $J(w^{[l]}, b^{[l]}) = \frac{1}{m}\sigma_{i=1}^{m} L(\hat{y^{(i)}}, h^{(i)}) + \frac{\lambda}{2m}\sigma_{l=1}^{L}||w^{[l]}||_F^2$
  - 그 결과로 간단하고 작은 신경망이 되기에 과대적합이 덜 일어납니다.

2. $tanh$ 활성화 함수를 사용했을 경우 λ 값 커지면 비용함수에 의해 w 는 작아지게 되고, 이때 $z^{[l]} = w^{[l]}a^{[l-1]} + b^{[l]}$ 이므로 z 도 작아지게 됩니다.

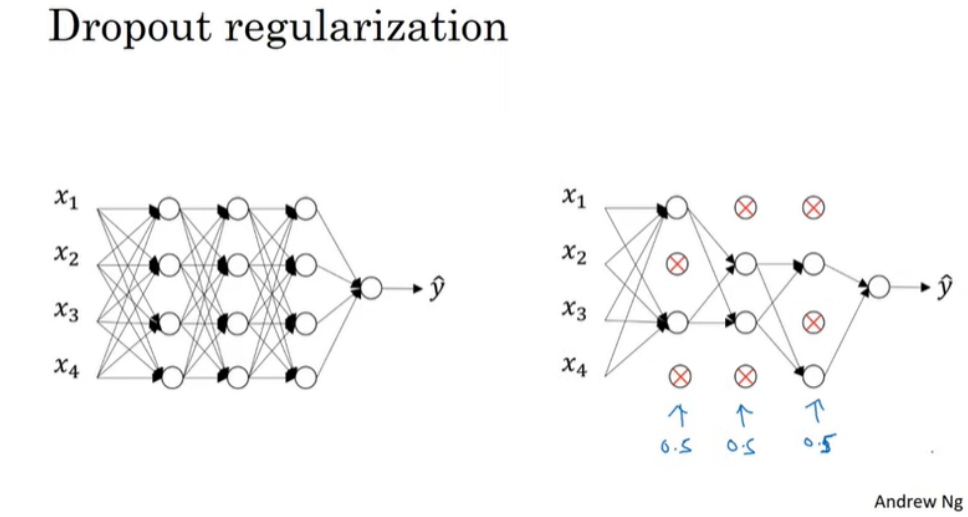


- 위의 그림을 보면 z 가 작을 때 $g(z)$ 는 선형 함수가 되고, 전체 네트워크도 선형이 되기에 과대적합과 같이 복잡한 결정을 내릴 수 없습니다.

📌 드롭아웃 정규화(C2W1L06)

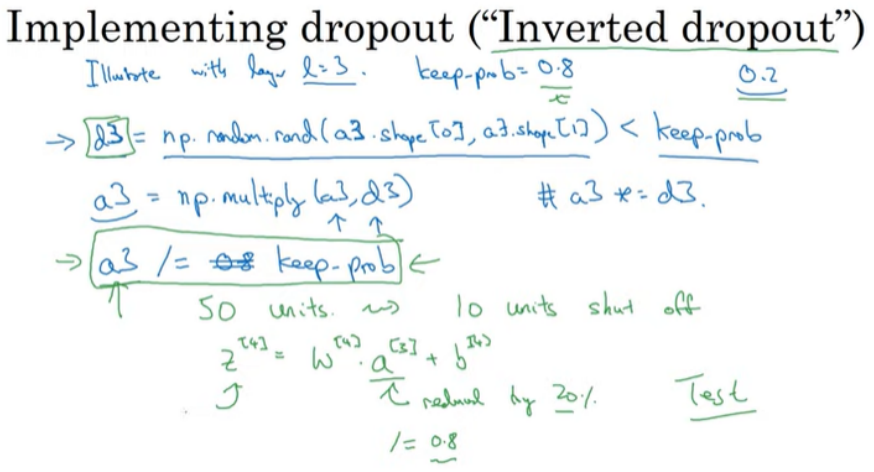
핵심어: 드롭아웃(Dropout), 역 드롭아웃(Inverted Dropout)

L2 정규화 외에 또 다른 매우 강력한 정규화 기법은 드롭아웃이다.



Andrew Ng

왼쪽처럼 과적합한 신경망을 훈련시킨다고 가정하자. 드롭아웃 방식은 신경망의 각각의 층에 대해 노드를 삭제하는 확률을 설정하는 것이다. 이 각각의 층에 대해, 각각의 노드마다 동전을 던지면 0.5의 확률로 해당 노드를 유지하고 0.5 확률로 노드를 삭제하게 된다. 동전을 던진 후 노드를 삭제했다고 했을 때, 그 후 삭제된 노드의 들어가는 링크와 나가는 링크를 모두 삭제한다. 그럼 더 작고 간소화된 네트워크가 되며, 이 감소된 네트워크에서 하나의 샘플을 역전파로 훈련시킨다. 다른 샘플에 대해서도 동전을 던지고 한 세트의 노드들을 남기며 다른 한 세트의 노드들은 완전히 삭제한다. 그럼 각각의 훈련 샘플에 대해서 감소된 네트워크를 사용해 훈련시키게 된다. 이것은 노드를 무작위로 삭제하는 이상한 기법처럼 보일 수 있는데, 실제로 잘 작동한다. 이것은 각각의 샘플에서 더 작은 네트워크를 훈련시키는 방식이며 이것이 네트워크를 정규화할 수 있는 것은 더 작은 네트워크를 훈련시키기 때문이다.



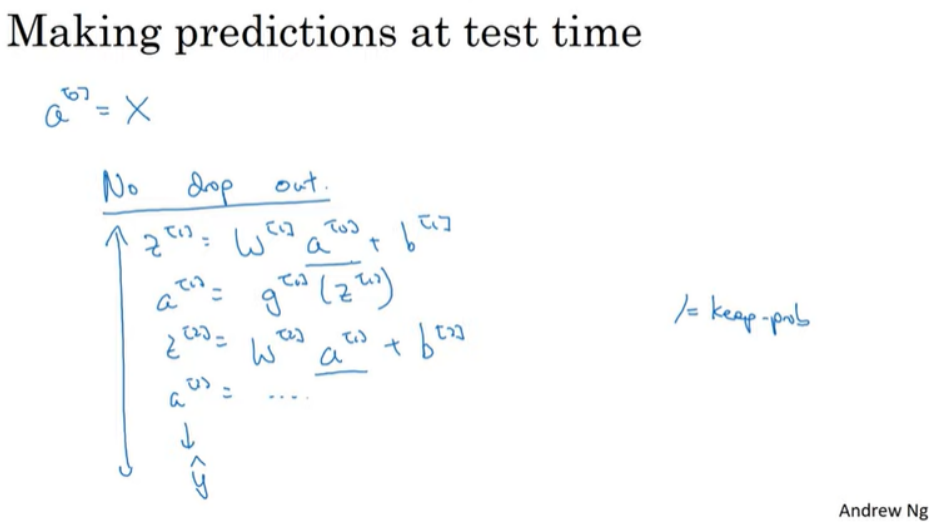
Andrew Ng

드롭아웃을 어떻게 구현하는지 살펴보자. 몇 가지 방법이 있는데, 역 드롭아웃이 가장 일반적이다. 예를 들어 층이 3인 경우에서 코드를 작성할 것이다(단일층에서 드롭아웃을 구현하는 방법). 먼저 층 3에 대한 드롭아웃 벡터인 벡터 d를 np.random.rand로 설정하며 이는 a3와 같은 모양을 가지게 된다. 그리고 이 값이 keep_prob이라는 숫자보다 작은지 비교하는 것이다. keep_prob의 값은 이전 슬

라이드에서는 0.5였지만 여기서는 0.8로 설정한다. 이 수는 주어진 은닉 유닛이 유지될 확률이며 예를 들어 keep_prob이 0.8이라는 것은 어떤 은닉 유닛이 삭제될 확률이 0.2라는 것이다. 이 코드는 무작위의 행렬을 생성하는데, 벡터화된 경우에도 잘 실행된다. 따라서 d3는 각각의 샘플과 각각의 은닉 유닛에 대해 0.8의 확률로 대응하는 d3가 1의 값을 가지고 0.2의 확률로 0의 값을 가지는 행렬이 될 것이다. 무작위의 숫자가 0.8보다 작을지의 여부는 0.8의 확률로 1이고(True), 0.2보다 작을지의 여부는 0.2의 확률로 0(False)가 된다.

다음은 3번째 층의 활성화인 a3에 관한 코드를 작성한다. a3의 값은 예전 a3의 값에 요소별 곱셈으로 d3을 곱해준 것이다(a3*=d3으로 써줘도 된다). 모든 원소에 대해 20퍼센트의 확률로 0이 되는 d3의 원소를 곱해 대응되는 a3의 원소를 0으로 만들게 된다. 이것을 파이썬으로 구현한다면 d3는 True와 False의 값을 갖는 boolean 타입의 행렬이 될 것이다. 그러나 0과 1이 아닌 boolean 타입의 값은 곱셈을 할 수 없게 되는데, 파이썬으로 구현하면 이를 알 수 있다. 최종적으로 얻은 a3를 0.8(keep_prob 매개변수)로 나눠준다. 예를 들어 세 번째 은닉 층에 50개의 유닛, 즉 50개의 뉴런이 있다고 가정하자. 그럼 a3는 (50,1) 차원일 것이며 벡터화한다면 (50, m)차원이 될 것이다. 80퍼센트의 확률로 유지하고 20퍼센트의 확률로 삭제한다면 평균적으로 10개의 유닛이 삭제, 즉 0의 값을 갖게 된다. z[4]의 값을 살펴보면 w[4]*a[3]+b[4]와 같은데, 예상대로 이 값은 20퍼센트만큼 줄어들 것이다. 즉 a[3] 원소의 20퍼센트가 0이 된다는 의미이다. z[4]의 기댓값을 줄이지 않기 위해 이 값을 0.8로 나눠줘야 하는데, 이는 필요한 20퍼센트 정도의 값을 다시 원래대로 만들 수 있기 때문이다. 이를 통해 a3의 기댓값을 유지할 수 있다. 다시 말해, a3/=keep_prob 식이 역 드롭아웃이라고 불리는 기법인 것이다. keep_prob을 0.8, 0.9, 1 등 어떤 값으로 설정하든(1이면 값이 그대로 유지되므로 드롭아웃이 없는 것), keep_prob을 다시 나눠줌으로써 a3의 기댓값을 같게 유지한다. 그리고 다음 슬라이드에서 살펴볼 신경망을 평가하는 경우에, 이 역 드롭아웃 기법이 테스트를 쉽게 만들어주며 이는 스케일링 문제가 적기 때문이다. 역 드롭아웃은 요즘 가장 보편적인 드롭아웃 기법이기도 하다.

여기서 우리는 d 벡터를 사용하여, 서로 다른 훈련 샘플마다 다른 은닉 유닛들을 0으로 만들게 된다. 같은 훈련 세트를 통해 여러 번 반복하면 그 반복마다 0이 되는 은닉 유닛은 무작위로 달라져야 한다. 따라서 하나의 샘플에서 계속 같은 은닉 유닛을 0으로 만드는 것이 아닌 경사하강법의 하나의 반복마다 0이 되는 은닉 유닛들이 달라진다. 세 번째 층의 d3 벡터는정방향과 역방향 전파 모두에서 어떤 노드를 0으로 만들지 결정한다.



알고리즘 훈련 후 테스트 시간에 하는 일을 설명하자면, 예측을 하고 싶은 x라는 샘플이 주어지고 이 테스트 샘플인 x를 표현하기 위해 0번째 층의 활성화 a[0]을 사용한다. 테스트에서는 드롭아웃을 사용하지 않는다. 그리고 z[1]을 w[1]a[0]+b[1], a[1]=g[1](z[1]), z[2]=w[2]a[1]+b[2], a[2]도 같은 방식으로 설정한다. 마지막 층에 이르면 y의 예측값을 얻는다. 그러나 테스트에서는 명시적으로 드롭아웃을 사용하지 않기 때문에 어떤 은닉 유닛을 삭제할지에 관한 동전을 무작위로 던지지 않는다. 그 이유는 테스트에서는 예측을 하는 것이므로 결과가 무작위로 나오는 것을 원하지 않는다. 테스트에 드롭아웃을 구현하는 것은 노이즈만 증가시킬 뿐이다. 이론적으로 무작위로 드롭아웃된 서로 다른 은닉 유닛을 예측 과정에서 여러 번 반복해 그들의 평균을 낼 수도 있지만, 컴퓨터적으로 비효율적이기도 하고 앞선 과정과 거의 비슷한 결과를 낸다. 이전 슬라이드에서 언급한 keep_prob으로 나누는 역 드롭아웃 효과는 테스트에서 드롭아웃을 구현하지 않아도 활성화 기댓값의 크기는 변하지 않기 때문에 테스트 할 때 스케일링 매개변수를 추가해주지 않아도 된다.

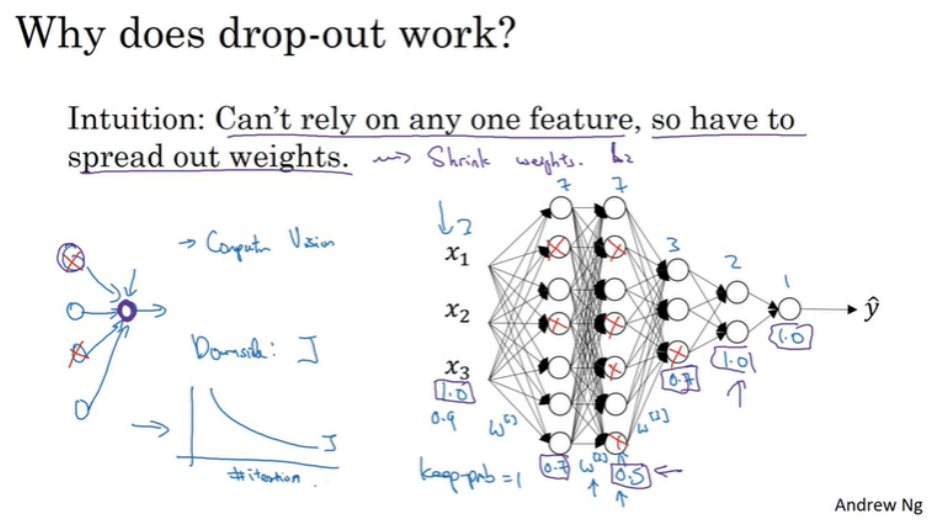
<정리>

- 드롭아웃의 방식은 신경망의 각각의 층에 대해 노드를 삭제하는 확률을 설정하는 것이다. 삭제할 노드를 랜덤으로 선정 후 삭제된 노드의 들어가는 링크와 나가는 링크를 모두 삭제한다.
- 그럼 더 작고 간소화된 네트워크가 만들어지고 이때 이 작아진 네트워크로 훈련을 진행하게 된다.
- 역 드롭아웃이라고 불리는 가장 일반적인 기법은 위와 같이 노드를 삭제후에 얻은 활성화 값에 keep.prop(삭제하지 않을 확률)을 나눠 주는 것이다.
- 이는 기존에 삭제하지 않았을 때 활성화 값의 기대값으로 맞춰주기 위함이다.

📌 드롭아웃의 이해(C2W1L07)

핵심어: 드롭아웃(Dropout)

드롭아웃은 무작위로 신경망의 유닛을 삭제시키는 ,이상하다고도 볼 수 있는 기법이다. 왜 이것이 정규화로 잘 작동하는 것인지 알아보도록 하자.



이전 강의에서 신경망의 유닛을 무작위로 삭제하는 것에 대한 설명을 했었는데, 따라서 모든 반복마다 더 작은 신경망에서 작업하게 된다. 더 작은 신경망을 사용하는 것이 정규화의 효과를 주는 것처럼 보인다. 여기 두 번째 직관이 있는데, 단일 유닛 관점에서 살펴보는 것이다. 이 단일 유닛이 해야 하는 일은 입력을 받아 의미 있는 출력을 생성하는 것이다. 드롭아웃을 통해 입력은 무작위로 삭제될 수 있다. 어떤 경우는 두 유닛이 삭제되고 어떤 경우에는 또 다른 유닛이 삭제된다. 따라서 보라색으로 표시한 유닛은 어떤 특성에도 의존할 수 없다. 그 특성이 무작위로 바뀌거나, 특성의 고유한 입력이 무작위로 바뀔 수 있기 때문이다. 따라서 이 특정 입력에 모든 것을 걸 수는 없는 상황이다. 즉 특정 입력에 유난히 큰 가중치를 부여하기가 꺼려지는 상황인 것이다. 그러므로 이 네 개의 입력 각각에 가중치를 분산시키는 편이 좋다고 볼 수 있다. 가중치를 분산시킴으로써 가중치의 norm의 제곱값이 줄어들게 된다. L2 정규화에서 봤던 것처럼 드롭아웃을 구현하는 효과는 가중치를 줄이는 것이고 L2 정규화처럼 과대적합을 막는데 도움이 된다. 드롭아웃은 예전에 L2 정규화의 적응형으로 보여지기도 했다. 그러나 L2 정규화에서 다른 가중치는 다르게 취급하는데, 그 가중치에 곱해지는 활성화의 크기에 따라 다르다. 정리하자면 드롭아웃은 L2 정규화와 비슷한 효과를 보여줄 수 있다. L2 정규화가 다른 가중치에 적용된다는 것과 서로 다른 크기의 입력에 더 잘 적응한다는 것만이 차이점이라고 할 수 있다.

드롭아웃을 구현할 때의 세부사항을 하나 더 추가하자면, 여기 3개의 입력 특성과 7개, 7개, 3개, 2개, 1개의 은닉 유닛이 있는 네트워크가 있다. 우리가 선택해야 할 매개변수 중에 keep_prob이라는 것이 있으며 각 층에 해당 유닛을 유지할 확률이다. 층마다 keep_prob을 바꾸는 것도 가능하다. 첫 번째 층의 가중치 행렬 w[1]은 (3, 7) 행렬, 두 번째 가중치 행렬은 (7, 7) 행렬, w[3]는 (7,3) 행렬, 이런 식으로 계속 진행된다. w[2]가 (7, 7)으로 가장 많은 매개변수를 갖기 때문에 가장 큰 가중치 행렬이다. 따라서 이 행렬의 과대적합을 줄이기 위해 층2는 상대적으로 낮은 keep_prob을 가져야 한다. 반면 과대적합의 우려가 적은 층에서는 더 높은 keep_prob을 지녀도

된다. 과대적합의 우려가 없는 층은 keep_prob을 1로 정해도 문제 없다. 보라색 박스를 친 숫자는 서로 다른 층에 대한 keep_prob의 값들이다. 1.0의 값을 갖는 keep_prob의 의미는 모든 유닛을 유지하고 해당 층에서는 드롭아웃을 사용하지 않는다는 의미이다. 그러나 매개변수가 많은 층, 즉 과대적합의 우려가 많은 층은 더 강력한 형태의 드롭아웃을 위해 keep_prob을 작게 설정한다. L2 정규화에서, 다른 층보다 더 많은 정규화가 필요한 층에서 매개변수 λ를 증가시키는 것과 비슷한 것이다. 이론적으로 드롭아웃을 입력 층에도 적용시킬 수 있으며 한 개나 그보다 더 많은 개수의 입력 특성을 삭제할 수도 있다. 그러나 이것을 실제로 자주 사용하지 않는 것이 좋으며 입력층에 대해서는 1.0에서 0.9의 keep_prob을 설정하는 것이 좋다. 입력 특성의 절반 이상을 삭제하는 일을 방지하기 위해서이다. 정리하자면 다른 층보다 과대적합의 우려가 더 큰 층에 대해서는 다른 층보다 더 낮은 값의 keep_prob을 설정할 수 있다. 단점은 교차 검증을 위해 더 많은 하이퍼파라미터가 생긴다는 부분이다. 또 다른 대안으로는 어떤 층에는 드롭아웃을 적용하고 어떤 층에는 적용하지 않아서 매개변수를 드롭아웃을 적용한 층에 대한 keep_prob 하나만 갖는 것이다.

이에 더하여, 컴퓨터 비전 분야에서 드롭아웃의 구현에 관한 최초의 성공들이 많이 나왔다. 컴퓨터 비전은 아주 많은 픽셀 값을 모두 사용하기 때문에 대부분의 경우 데이터가 부족하다. 따라서 컴퓨터 비전에서 드롭아웃이 매우 빈번하게 사용되며, 최근 비전 분야의 연구원들은 거의 항상 기본값으로 드롭아웃을 사용한다. 그러나 여기서 기억해야 할 것은 드롭아웃은 정규화 기법이고 과대적합을 막는 데 도움을 준다는 것이다. 따라서 자신의 네트워크가 과대적합의 문제가 생기기 전까지는 드롭아웃을 사용하지 않는 것이 좋다(어떤 애플리케이션 영역에서는 자주 사용되는 한이 있더라도). 컴퓨터 비전은 충분한 데이터가 없기 때문에 거의 대부분 과대적합이 일어나고 드롭아웃을 많이 사용하는 이유이다. 이러한 일이 다른 분야에도 항상 일반화 되는 것은 아니다. 드롭아웃의 큰 단점은 비용함수 J가 더 이상 잘 정의되지 않는다는 것이다. 모든 반복마다 무작위로 한 뭉치의 노드들이 삭제되는데, 따라서 경사 하강법의 성능을 이중으로 확인한다면 모든 반복에서 잘 정의된 비용함수 J가 하강하는지 확인하는 것이 어려워지며, 이로 인해 최적화된 비용함수가 잘 정의되지 않아 계산이 어려워지는 것이다. 보통은 keep_prob을 1로 설정해서 드롭아웃 효과를 멈추고, 코드를 실행시켜 J가 단조감소하는지 확인한다. 그리고 드롭아웃 효과를 다시 주고 드롭아웃이 있을 때 코드를 바꾸지 않도록 한다. 그 이유는 드롭아웃이 있을 때 코드와 경사하강법이 잘 작동하는지 함수만 보고 확인하는 것 외에 다른 방법이 필요하기 때문이다.

<정리>

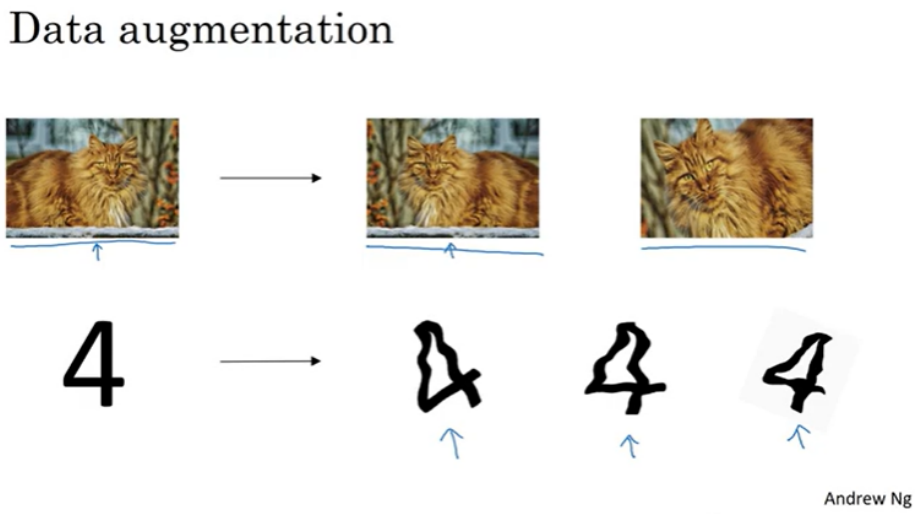
- 드롭아웃은 랜덤으로 노드를 삭제 시키기 때문에, 하나의 특성에 의존 하지 못하게 만듦으로서 가중치를 다른 곳으로 분산 시키는 효과가 있다.
- 드롭아웃의 keep_prop 확률은 층마다 다르게 설정 할 수 있다.
- 모든 반복에서 잘 정의된 비용함수가 하강하는지 확인하는게 어려워진다. 따라서 우선 드롭아웃을 사용하지 않고, 비용함수가 단조감소인지 확인 후에 사

용해야 한다.

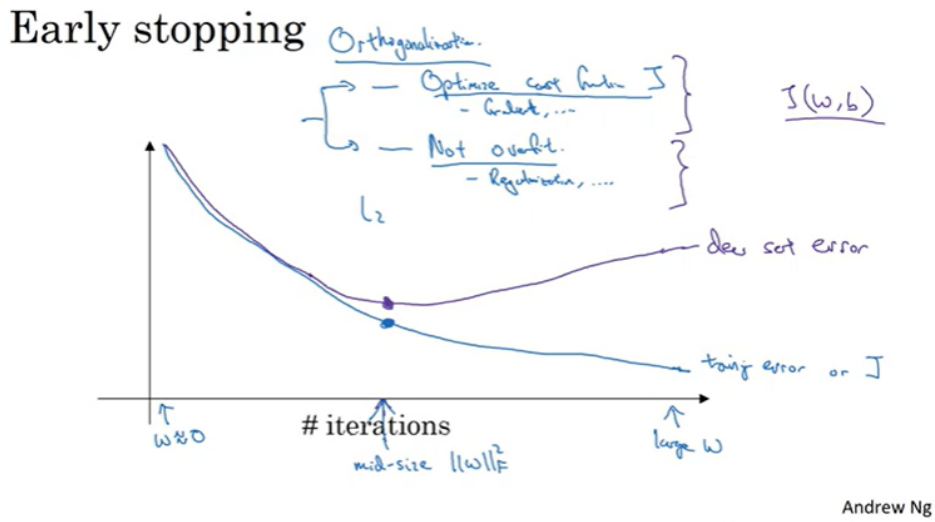
📌다른 정규화 방법들(C2W1L08)

핵심어: 데이터증식(Data augmentation), 조기종료(Early stopping)

L2 정규화와 드롭아웃 정규화와 더불어 신경망의 과대적합을 줄이는 다른 기법들을 알아보자.



고양이 분류기를 훈련시키는 경우, 더 많은 훈련 데이터가 과대 적합을 해결하는 데 도움을 줄 수 있지만 많은 비용이 들어가거나 불가능한 경우가 있다. 그러나 수평 방향으로 뒤집은 이미지를 훈련 세트에 추가시켜 훈련 세트를 늘리는 방법도 있다. 즉 오른쪽의 이미지도 샘플에 추가하는 것이다. 따라서 이미지를 수평 방향으로 뒤집어 훈련 세트를 두 배로 증가시킬수 있다. 새로운 m개의 독립적은 샘플을 얻는 것보다 이 방법은 중복된 샘플들이 많아져서 좋지는 않지만, 새로운 고양이 사진을 더 많이 구하지 않고 할 수 있는 방법이다. 이 방법 외에도 무작위로 이미지를 편집해 새로운 샘플을 얻을 수도 있다. 예를 들어 이미지를 회전시키고 무작위로 확대시킬 수 있다. 이는 여전히 고양이 데이터이며, 이러한 이미지의 무작위적인 왜곡과 변형을 통해 데이터 세트를 증가시키고 추가적인 가짜 훈련 샘플을 얻을 수 있다. 하지만 이런 추가적인 가짜 이미지들은 완전히 새로운 독립적인 고양이 샘플을 얻는 것보다 더 많은 정보를 추가해주지는 않을 것이다. 다만 컴퓨터적인 비용이 들지 않고 할 수 있다는, 즉 데이터를 더 얻을 수있는 비싸지 않은 방법이라는 장점이 있다. 이 방법을 통해 정규화를 하고 과대적합을 줄일 수 있다. 이렇게 합성한 이미지를 통해 사용하는 알고리즘에게 고양이 이미지를 뒤집어도, 확대해도 여전히 고양이라는 것을 학습시킬 수 있다. 한편 시각적인 문자 인식의 경우, 마찬가지로 숫자를 얻어 무작위의 회전과 왜곡을 부여할 수 있다. 이것들을 훈련 세트에 추가해도 여전히 숫자 4를 나타낸다. 따라서 데이터 증가는 정규화 기법과 비슷하게 사용될 수 있다.



조기종료라고 부르는 또 다른 기법이 있다. 경사 하강법을 실행하면서 훈련 세트에 대한 분류 오차, 즉 훈련 오차를 그리거나 최적화하는 비용함수 J를 그리게 된다. 훈련 오차나 비용함수 J는 위 그래프와 같이 단조 감소하는 형태로 그려져야 한다. 조기종료에는 개발 세트 오차도 함께 그려주는데, 이것은 개발 세트의 분류 오차 혹은 개발 세트에서 평가되는 로지스틱 손실 함수라고 볼 수 있다. 여기서 발견할 수 있는 것은 개발 세트 오차가 아래로 내려가다가 특정 부분에서 증가한다는 점이다. 조기 종료는 신경망이 이 반복 주변에서 가장 잘 작동하는 것을 알 수 있으며, 따라서 중간에 신경망을 훈련시키는 것을 멈추고 이 개발 세트 오차를 만든 값을 최적으로 삼는다. 신경망에서 많은 반복을 실행시키지 않은 경우 매개변수 w는 0에 가깝다. 무작위의 작은 값으로 초기화시켜 오랜 시간 훈련시키기 전까지 w의 값은 여전히 작다. 반복을 실행할수록 w의 값은 계속 커지며, 이로 인해 이 지점(거의 맨 오른쪽)에서는 매개변수 w의 값이 훨씬 커진 상태이다. 조기 종료 기법에서 반복을 중간에 멈추면 w가 중간 크기의값을 갖는 상태이다. L2 정규화와 비슷하게 매개변수 w에 대해 더 작은 norm을 갖는 신경망을 선택함으로써 신경망이 덜 과대적합하게 되는데, 이후 조기종료라는 말 뜻 그대로 신경망을 조기로 종료한다. 신경망을 훈련시킬 때 가끔 조기종료를 사용하는데, 여기에는 하나의 단점이 있다.

머신러닝 과정은 서로 다른 몇 가지 단계로 이루어져 있으며 첫 번째로 비용함수 J를 최적화하는 알고리즘을 원한다. 경사하강법처럼 이를 위한 몇 가지 알고리즘이 있는데, 모맨텀, RMSProp, Adam 등이 있다. 비용함수 J를 최적화하고 난 뒤에 과대적합되는 것을 막기 위한 몇가지 도구들이 또 있는데, 이는 정규화, 데이터 더 추가하기 등의 방법이다. 머신러닝에서 이미 아주 많은 하이퍼파라미터들이 있고 여러 가능한 알고리즘 중 선택하는 것은 매우 복잡하기 때문에, 비용함수 J를 최적화하는 하나의 도구 세트만 있다면 머신러닝이 훨씬 더 간단해질 수 있다. 비용함수 J를 최적화할 때 집중하는 것은 w와 b를 찾는 것인데, J(w, b)가 가능한 작아지는 값을 찾는 것 외에는 신경쓰지 않는다. 이는 과대적합을 막는 것, 즉 분산을 줄이는 것은 완전히 다른 일이며 이를 위한 별개의 도구들이 필요하다. 이 원리는 직교화라고 부르기도 한다. 한 번에 하나의 할 일만을 생각하는 것이다.

여기서 조기 종료의 주된 단점은, 이 둘을 섞어버린다는 것이다. 이 두 문제(비용함수 J 최적화 & 과대적합 막기)에 대해 더 이상 독립적으로 작업할 수 없게 되는 것이다. 경사하강법을 일찍 멈춤으로써 비용함수 J를 최적화하는 것을 멈추게 되고, 비용함수 J를 줄이는 일을 잘 해내지 못하게 되며 동시에 과대적합을 막으려고 하는 것이다. 따라서 두 문제를 해결하기 위해 서로 다른 도구를 사용하는 대신 혼합된 하나의 도구를 사용하게 되는 것이다. 그리고 이것은 문제를 더 복잡하게 만든다. 조기 종료를 사용하는 것의 대안은 L2 정규화를 사용하는 것이며, 그럼 가능한 오래 신경망을 훈련시킬 수 있게 된다. 이를 통해 하이퍼파라미터의 탐색 공간이 더 분해하기 쉽고 찾기 쉬워진다. 그러나 단점은 정규화 매개변수 λ에 많은 값을 시도해야 한다는 것이다. λ의 많은 값을 대입하는 것은 컴퓨터적으로 많은 비용이 든다. 조기 종료의 진짜 장점은 경사하강법 과정을 한 번만 실행하여, L2 정규화의 하이퍼파라미터 λ와 같이 많은 값을 얻을 필요없이 작은 w, 중간 w, 큰 w의 값을 얻게 되는 것이다. 앞서 설명한 단점에도 불구하고 많은 사람들이 조기 종료 기법을 사용하며, 컴퓨터적으로 감당할 수 있다면 L2 정규화를 사용해 λ에 많은 값을 시도하는 것도 좋은 방법이다. 그러나 조기 종료는 명시적으로 많은 값을 λ에 시도하지 않고도 비슷한 효과를 가져올 수 있다.

<정리>

- 데이터증식
  - 이미지의 경우 더 많은 훈련 데이터를 사용함으로서 과대적합을 해결 할 수 있습니다.
  - 보통 이미지를 대칭, 확대, 왜곡 혹은 회전 시켜서 새로운 훈련 데이터를 만듭니다.
  - 이런 추가적인 가짜 이미지들은 완전히 새로운 독립적인 샘플을 얻는 것보다 더 많은 정보를 추가해주지는 않지만, 컴퓨터적인 비용이 들지 않고 할 수 있다는 장점이 있습니다
- 조기종료
  - 훈련세트의 오차는 단조하강함수로 그려질 것입니다.
  - 조기종료에서는 개발 세트의 오차도 그려줍니다.
  - 만약에 개발세트의 오차가 어느 순간 부터 하락 하지 않고 증가하기 시작하는 것이라면 과대적화가 되는 시점입니다.
  - 따라서, 조기 종료는 신경망이 개발 세트의 오차 저점 부근, 즉 가장 잘 작동하는 점일때 훈련을 멈추는 것입니다.
  - 단점: 훈련시 훈련 목적인 비용 함수를 최적화 시키는 작업과 과대적합하지 않게 만드는 작업이 있습니다. 두 작업은 별개의 일이라서 두 개의 다른 방법으로 접근해야 합니다. 그러나 조기 종료 두 가지를 섞어 버립니다. 이 따라서 최적의 조건을 찾지 못할 수도 있습니다.

공감

'**DL 스터디&프로젝트**' 카테고리의 다른 글

[Euron 중급 세션 5주차] 딥러닝 2단계 1. 머신러닝 어플리케이션 설정하기 (0) 2023.10.09
[Euron 중급 세션 4주차] 5. 심층 신경망 네트워크 (1) 2023.10.02
[Euron 중급 세션 3주차] 4. 얕은 신경망 네트워크 (0) 2023.09.25
[Euron 중급 세션 2주차] 3. 파이썬과 벡터화 (0) 2023.09.18