Fundamentals of Deep Learning

Chapter 4. 경사 하강법을 넘어서

4.1. 경사 하강법의 과제

신경망의 배경을 이루는 기본 개념은 지난 수십 년 동안 존재해 왔지만, 최근까지도 신경망 기반 학습 모델이 주류가 되지는 못했음. 신경망이 지닌 매력은 여러 층으로 신경망을 생성함으로써 열리게 된 특성 즉, 신경망의 표현력과 관련이 있음. 앞 장들에서 논의한 것처럼, 심층 신경망은 지금까지 다루기가 매우 까다롭다고 여겨지던 문제들을 해결할 수 있음. 그러나 심층 신경망을 끝에서 끝까지 학습시키는 것은 방대하게 분류된 데이터셋과 GPU 가속의 형태로 더 향상된 하드웨어, 여러 가지 알고리즘적 발경 등을 포함한 많은 기술 혁신이 요구되는 어려운 도전 과제들로 가득 차 있음.

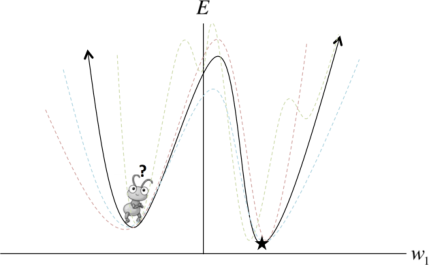
여러 해 동안, 연구자들은 딥러닝 모델로 표현되는 복잡한 오차 곡면들을 극복하기 위해 탐욕적 층별 학습에 의존했음. 이 시간 집약적인 전략들은 최적 파라미터 설정으로 수렴하는 미니배치 경사 하강법을 사용하기 전에 한 번에 한 층씩 모델 파라미터들의 더 정확한 초기값을 찾으려 했음. 하지만 최근에 최적화 방법에 대한 돌파구가 마련되면서 끝에서 끝까지 방식으로 모델을 직접 학습시키는 것이 가능해짐.

이번 장에서는 이 돌파구를 논의함. 다음 절에서는 주로 지역 최솟값에 초점을 맞추고 이것이 성공적으로 심층 모델들을 학습시키는 데 장애가 되는지를 살펴봄. 이어서 심층 모델로부터 유도된 비볼록 오차 곡면들과 기본 미니배치 경사 하강법이 왜 짧게 떨어지는지와 최신 비볼록 최적화 도구가 어떻게 이 함정들을 극복하는지를 좀 더 탐색해 봄.

4.2. 심층 신경망의 오차 곡면에서 지역 최솟값

딥러닝 모델 최적화의 주요 과제는 오차 곡면의 전역 구조를 유츄하기 위해 최소한의 지역 정보를 사용해야 한다는 점임. 이것은 일반적으로 지역과 전역 구조 사이의 유사성이 거의 없어서 매우 어려운 문제임. 다음 비유를 예로 들어보자. 당신이 미국 땅 위에 있는 개미라고 가정해보자. 당신은 지도상에 무작위로 떨어짐. 당신의 목표는 이 표면에서 가장 낮은 지점을 찾아가는 것임. 어떻게 해야 할까? 관찰할 수 있는 것이 당신을 둘러싸고 있는 매우 가까운 주변뿐이라면 이것은 매우 다루기 어려운 문제처럼 보임. 미국의 지표면이 오목한 사발 모양이고 학습률을 적절하게 잘 정한다면, 경사 하강 알고리즘을 사용해 결국에는 사발의 밑바닥 지점을 찾아낼 수 있을 것임.

그러나 미국의 지표면은 매우 복잡함. 즉, 표면이 볼록하지 않음. 이것은 설사 계곡을 찾는 데 성공할지라도 그것이 지도에서 가장 얕은 계곡인지는 알 도리가 없다는 것을 의미함. 2장에서 0의 경사를 갖은 최소인 듯한 가짜 지역에 있을 때, 미니배치 경사 하강법이 이 다루기 힘든 오차 곡면을 탐색하는 데 어떻게 도움을 줄 수 있는지를 이야기했음. 하지만 밑의 그림을 보면 확률적 오차 곡면조차도 깊은 지역 최솟값으로부터 우리를 구해낼 수 없음.

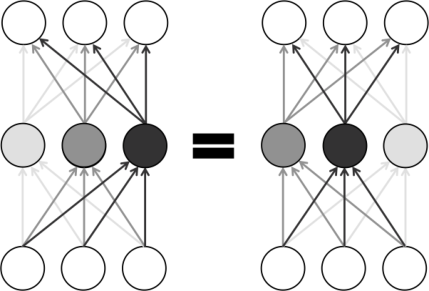


이제 아주 중요한 질문에 직면하게 됨. 이론적으로 지역 최소값은 중요한 문제를 제기함. 하지만 실제로 심층 신경망의 오차 곡면에서 지역 최소값은 얼마나 일반적인 것일까? 그리고 실제로 어떤 시나리오에서 학습상 문제가 발생하는가? 이어지는 두 절에서 지역 최소값에 대한 일반적인 오해를 다루어보겠음.

4.3. 모델 식별성

지역 최소값의 첫 번째 근원은 일반적으로 모델 식별성이라는 개념과 관련 있음. 심층 신경망에 관해 한 가지 알려진 점은 오차 곡면이 아주 많은 수의 지역 최소값을 갖는 것이 확실하다는 것임. 여기에는 크게 두 가지 이유가 있음.

첫 번째는 완전연결 전방향 신경망의 한 층에서 뉴런들을 어떻게 재정렬해도 여전히 신경망의 끝에서는 똑같은 최종 출력이 나오기 때문임. 밑의 그림은 단순한 3개 뉴런으로 구성된 층을 예로 들었음. 결과적으로 n개 뉴런으로 구성된 한 층 안에서 파라미터를 재배열하는 방법은 n!개가 있음. 그리고 각각 n개 뉴런으로 구성된 l개 층의 심층 신경망은 설정이 총 개가 됨.



뉴런 재정렬의 대칭성 외에도 비식별성이 특정 종류의 신경망에서 다른 형태로 존재함. 예를 들어, 동일한 신경망에서 개별 ReLU 뉴런 결과를 갖는 무한한 수의 동일한 구성이 있다고 하자. ReLU는 조각마다 선형함수를 사용하므로 신경망의 동작을 바꾸지 않고도 0이 아닌 상수 k를 들어오는 모든 가중치에 곱하고, 나가는 모든 가중치에 1/k를 곱해 조정할 수 있음. 이것이 맞는지는 독자들이 연습을 통해 터득하기 바람.

그러나 궁극적으로 심층 신경망의 비식별성 때문에 발생한 지역 최소값들은 본질에서 문제가 되지 않음. 이것은 식별 불가능한 모든 구성이 제공되는 입력 값과 상관없이 구별할 수 없는 방식으로 작동하기 때문임. 이는 학습과 검증, 테스트 데이터셋 모두 동일한 오차를 발생시킨다는 뜻임. 다시 말해, 이 모든 모델은 학습 데이터로부터 똑같이 배우고, 본 적 없던 예제들에서도 일반화 과정을 통해 똑같이 동작할 것임.

그 대신 지역 최소값들이 가짜일 때만 문제가 됨. 가짜 지역 최소값은 전역 최소값 구성보다 더 큰 오차가 발생하는 신경망의 가중치 구성에 해당함. 이러한 종류의 지역 최소값이 일반적이라면 경사 기반 최적화를 이용할 경우 지역 구조만 고려할 수 있어서 금방 문제에 직면하게 됨.

4.4. 심층 신경망에서 가짜 지역 최소값들은 얼마나 다루기 어려운가?

수년 동안 딥러닝 전문가들은 근거는 빈약했지만, 그들의 모든 문제가 가짜 지역 최소값에서 심층 신경망을 학습시킨 데에서 비롯된다고 비난했음. 오늘날, 실제 심층 신경망에서 가짜 지역 최소값이 전역 최소값보다 상대적으로 높은 오차율을 보이는 것이 일반적인지 여전히 궁금증으로 남아 있음. 그러나 최근 많은 연구에서 지역 최소값 대부분이 전역 최소값과 매우 유사한 오차율과 일반화 특성을 나타냄을 보여주고 있음.

이 문제를 단수화하는 방법의 하나는 심층 신경망을 학습시킬 때 오차 함수의 값을 시간에 대해 그리는 것임. 그러나 이 전략은 오차 곡면이 울퉁불퉁한지 또는 나아갈 방향을 파악하는 데 어려움이 있는지를 알 수 없어서 오차 곡면에 대한 충분한 정보를 주지 못함.

이 문제를 더 효과적으로 분석하기 위해 굿펠로 등은 2014년에 이 두 잠재적 교란 요소의 분리를 시도하는 한 논문을 발표했음. 시간에 대한 오차 함수를 분석하는 대신에 선형 보간법을 사용해 임의로 초기화된 파라미터 벡터와 성공적인 최종 해 사이의 오차 곡면에서 어떤 일이 일어나는지를 조사했음. 그래서 임의로 초기화된 파라미터 벡터 와 확률적 경사 하강법의 해인 가 주어진 상태에서 선형 보간법

에 따라 모든 지점에서 오차 함수를 계산하는 것을 목표로 했음.

다시 말해, 어느 방향으로 나아갈지 뿐만 아니라 지역 최소값들이 경사 기반 탐색방법을 방해하는지도 조사히길 원했음. 이들은 다른 유형의 뉴런을 가진 실제 신경망의 폭넓은 다양성에도 불구하고, 파라미터 공간에서 임의로 초기화된 지점과 SGD의 해 사이의 직접 경로는 다루기 어려운 지역 최소값들로부터 영향을 받지 않는다는 것을 보였음.

3장에서 작성한 전방향 ReLU 신경망을 사용해 이것을 증명할 수 있음. 다음과 같이 전방향 신경망을 학습시키는 동안 저장한 체크포인트 파일로 추론 함수와 손실 함수를 다시 인스턴스화할 수 있고, 동시에 var\_list\_opt에서 나중에 사용할 수 있게 원래 그래프에서 변수들의 포인터 목록을 관리할 수 있음.

# 28\*29=784 형태의 MNIST 데이터 이미지  
x = tf.placeholder('float', [None, 784])  
# 0~9 숫자 인식 => 10 클래스  
y = tf.placeholder('float', [None, 10])  
  
sess = tf.Session()  
  
with tf.variable\_scope("mlp\_model") as scope:  
 output\_opt = inference(x)  
 cost\_opt = loss(output\_opt, y)  
 saver = tf.train.Saver()  
 scope.reuse\_variables()  
 var\_list\_opt = [  
 'hidden\_1/W',  
 'hidden\_1/b',  
 'hidden\_2/W',  
 'hidden\_2/b',  
 'output/W',  
 'output/b',  
 ]  
 var\_list\_opt = [tf.get\_variable(v) for v in var\_list\_opt]  
 saver.restore(sess, 'mlp\_logs/model-checkpoint-file')

이와 비슷하게, 임의로 초기화된 신경망을 생성하는 데 컴포넌트 생성자를 재사용할 수 있음. 여기서 프로그램의 다음 단계를 위해 다음과 같이 var\_list\_rand에 변수들을 저장함.