챕터8. 심층 모형의 훈련을 위한 최적화 기법

2023.03.28.

김은채

들어가기 전, 손실 함수와 목적 함수

설계된 머신러닝 모델은 어떤 임무 수행

→ 이것을 잘 했는지 못 했는지 판별

→ '잘 했다' 라는 정의의 경계가 모호

→ 모호함을 피하기 위해 손실함수(J) 정의 ('못 한 정도'를 0이 되도록 함.)

8.1 학습과 순수한 최적화의 차이점

$$J(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},\mathrm{y}) \sim \hat{p}_{\text{자료}}} L(f(\boldsymbol{x};\boldsymbol{\theta}), y).$$

$$J^*(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},\mathrm{y}) \sim p_{\text{자료}}} L(f(\boldsymbol{x};\boldsymbol{\theta}), y).$$

8.1.1 경험적 위험도 최소화

위험도(risk): 기대(평균) 일반화 오차

순수한 최적화: 모든 데이터를 알고 있는 경우 위험도 최소화는
최적화 알고리즘으로 풀 수 있는 최적화 문제

학습: 주어진 한정 된 훈련 집합만 있다면 기계학습 문제

8.1.1 경험적 위험도 최소화

기계학습에 의한 위험도 최소화(경험적 위험도 최소화):

$$\mathbb{E}_{\boldsymbol{x},\mathrm{y}\sim \hat{p}_{\text{자료}}(\boldsymbol{x},y)}[L(f(\boldsymbol{x};\boldsymbol{\theta}),y)] = \frac{1}{m}\sum_{i=1}^{m} L(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}),y^{(i)}).$$

- $p(\boldsymbol{x},y)$를 훈련집합으로 정의 된 $\hat{p}(\boldsymbol{x},y)$로 대체하는 것이 해당함
- 우리가 최소화 할 것은 경험적 위험도
- m은 훈련 견본 개수

8.1.1 경험적 위험도 최소화

문제점1. 과대적합이 일어나기 쉬움.

→ 모델의 유연성이 과도하게 높을 때도 발생하지만 훈련 데이터가 적을 때도 발생.

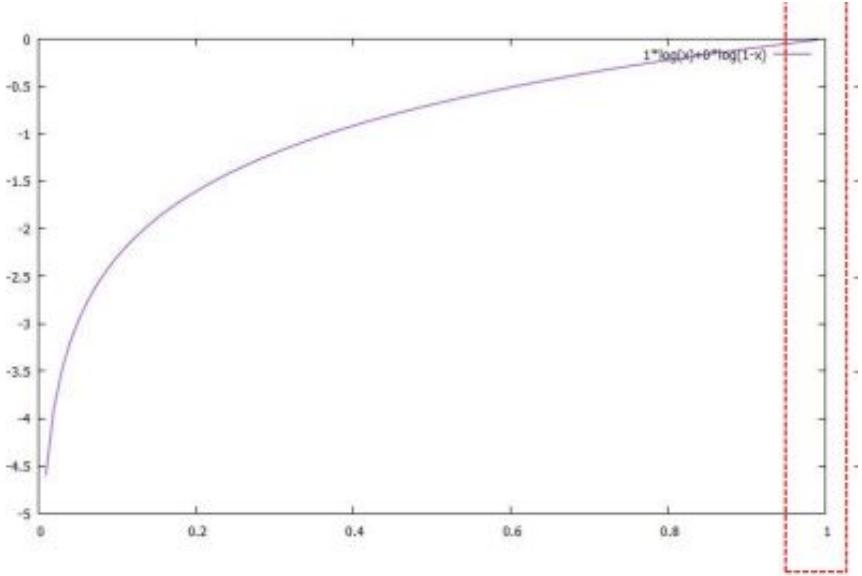
경험적 위험도는 한정된 훈련 데이터의 분포로 정의하기에 과적합 발생 여지 있음

문제점2. 현대 수치 최적화 기법은 경사 하강법을 기반으로 함

→ 목적함수의 기울기가 0이거나 도함수를 만들 수 없다면 소용 없음

예시) 0-1손실함수(Cross Entropy)

Cross Entropy

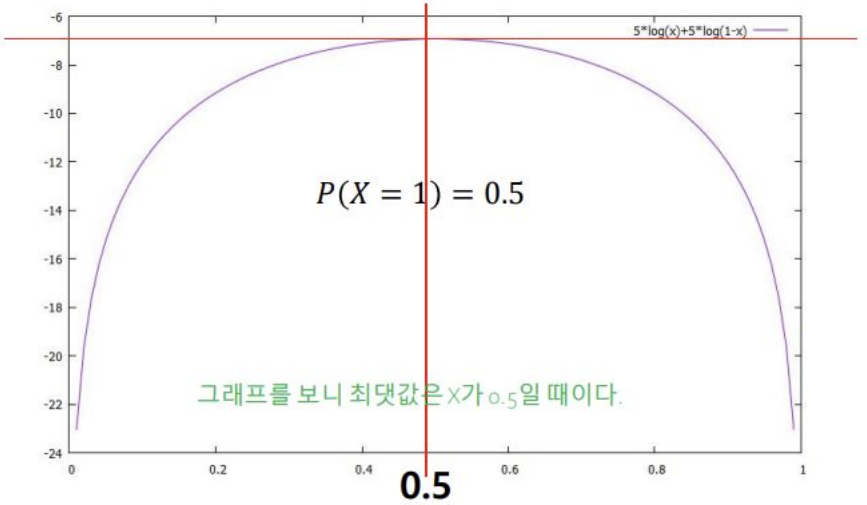


성공만 나온 경우

$P(X = 1) = 1$

1

성공과 실패가 반반 나온 경우



$P(X = 1) = 0.5$

그래프를 보니 최댓값은 x가 0.5일 때이다.

0.5

8.1.2 대리 손실함수와 조기 종료

경우에 따라 손실함수를 사용하여 모형의 학습 능력이 개선되기도 함

ex) 0-1손실 함수의 대리로 정확한 부류의 ‘로그 가능도’ 사용

8.1.2 대리 손실함수와 조기 종료

“훈련 알고리즘은 극소점에서 멈추지 않는다”

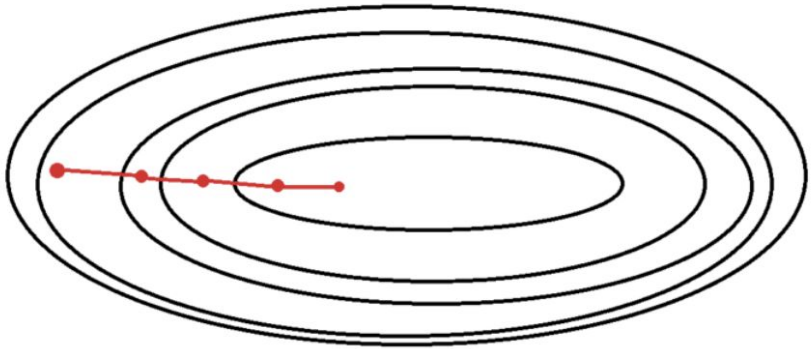
기계학습: 대리 손실 함수 최소화, 과대 적합 발생 시작 되면
조기종료 실행(종료판정 기준은 진 손실함수에 기초함)

순수한 최적화: 기울기가 작아지면 수렴이 일어난 것으로 간주,
실행종료

8.1.3 배치 알고리즘과 미니배치 알고리즘

배치 경사 하강법 = 결정론적 경사 하강법:

모든 훈련 견본을 하나의 커다란 배치(일광 처리 묶음)에서 동시에 처리함



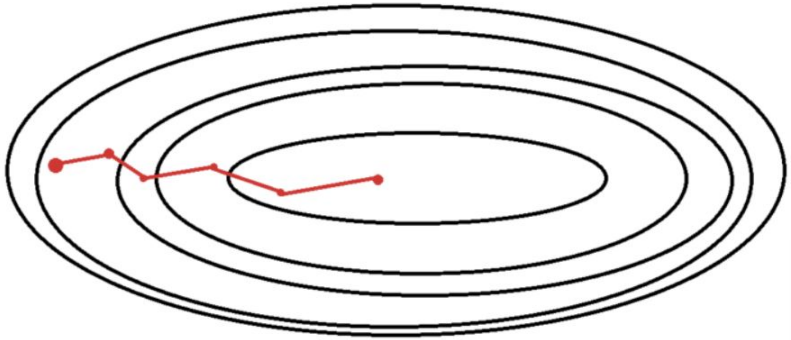
BGD의 시각화

- 전체 데이터를 통해 학습시키기 때문에, 가장 업데이트 횟수가 적다.
- 전체 데이터를 모두 한 번에 처리하기 때문에, 메모리가 가장 많이 필요하다.
- 항상 같은 데이터 (전체 데이터)에 대해 경사를 구하기 때문에, 수렴이 안정적이다.

8.1.3 배치 알고리즘과 미니배치 알고리즘

미니 배치=미니배치 확률적=확률적 방법:

전체 데이터를 batch_size개씩 나눠 배치로 학습(배치 크기를 사용자가 지정)



MSGD의 시각화

- 일부 하드웨어는 배열의 크기가 특정 값일 때 좋은 성능을 냄.
- 배치 알고리즘보다 계산량이 적음

8.2 신경망 최적화의 난제들

- 불량 조건/4.3.1 참고
- 극솟값
- 대지, 안장점, 기타 평평한 영역들
- 절벽과 기울기 폭발
- 장기 의존성
- 부정확한 기울기
- 국소 구조와 전역 주고의 부실한 대응 관계
- 최적화의 이론적 한계들

8.3 기본 알고리즘

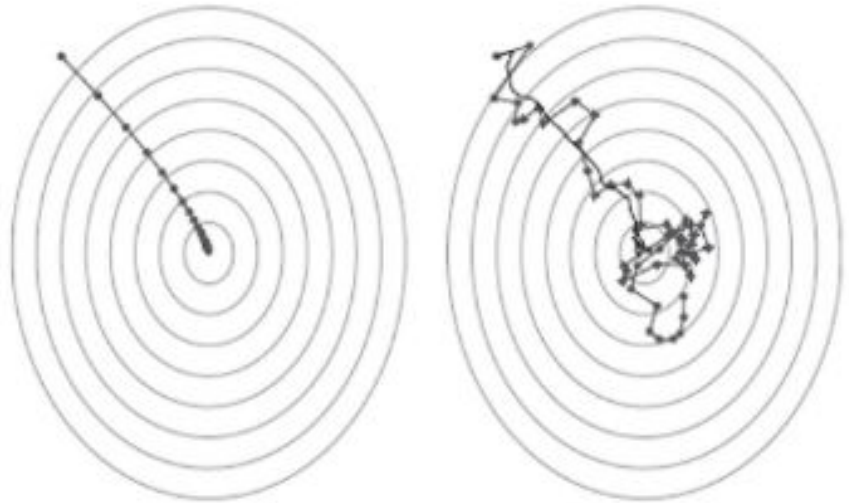
8.3.1 확률적 경사 하강법

8.3.2 운동량

8.3.3 네스테로프 운동량

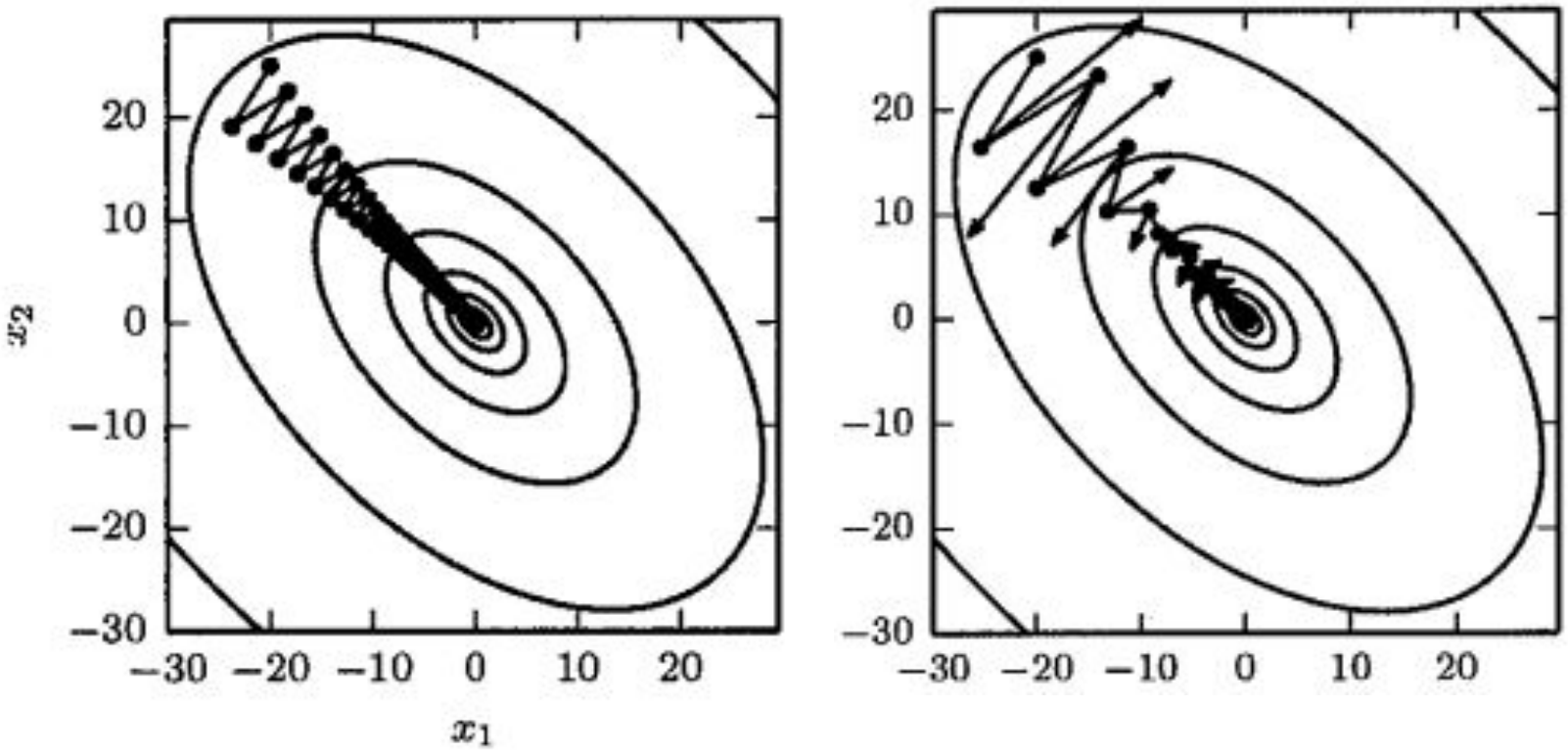
8.3.1 확률적 경사 하강법 SGD알고리즘

- 전체 데이터 중 단 하나의 데이터를 이용하여 경사 하강법을 1회 진행
- 부정확 할 수는 있지만 계산 속도가 훨씬 빠름→같은 시간 더 많은 step
- 요동치기에 지역 최소값에 빠져도 쉽게 나올 수 있지만 전역 최소값에 다다르기 힘듦

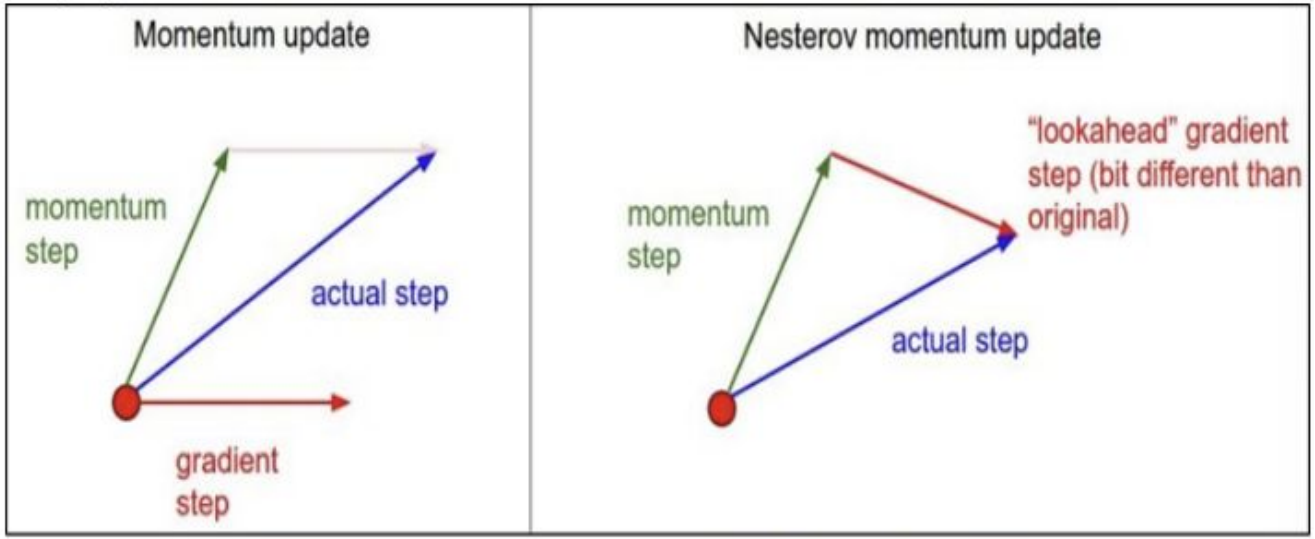


<BGD, SGD>

8.3.2 운동량



8.3.3 네스테로프 운동량



Difference between Momentum and NAG. Picture from CS231.

- 한 단계를 미리 예측함으로써 불필요한 이동을 줄임
- 오차의 수렴 속도가 O(1/k)에서 O(1/k^2)으로 올라감을 보여줌

8.4 매개변수 초기화 전략

<시작점을 어떻게 초기화 하느냐에 강하게 영향을 받음>

- 시작점에 따라 알고리즘이 수렴 할 수도 있고 실패할 수도 있음
- 학습이 수렴하는 경우에도 시작점에 따라 수렴이 빠를 수도 있고 느릴 수도 있음
- 수렴할 때 비용이 큰 점으로 수렴할 수도 있고 작은 점으로 수렴할 수도 있음
- 시작점은 일반화 오차에도 영향을 미침

8.4

그람 슈미트 직교화: 주어진 백터들을 이용해서 서로 수직인 백터를 만드는 방법 (직교기저 또는 정규 직교 기저를 구하는 과정)

정규화된 초기화: 모든 층의 분산이 같아지도록 초기화 한다는 목표와 모든 층의 기울기 분산이 같아지도록 초기화 한다는 목표를 갖고 고안 됨

희소 초기화: 가중치들을 초기화 할 때 각 단위에서 0이 아닌 가중치가 정확히 k개 이어야한다는 제약을 가함. 초기화 시점에서 단위들이 좀 더 다양해지게 하는데 도움 됨

8.5 학습 속도를 적절히 변경하는 알고리즘들

“만일 손실함수의 주어진 모형 매개변수에 대한 편미분의 부호가 바뀌지 않았다면, 학습 속도를 증가하고, 부호가 바뀌었다면 학습 속도를 감소해야한다”

라는 간단한 기초에 착안

8.5.1 AdaGrad

모든 단계의 기울기들을 누적한 값에 반비례하여 개별적으로 적응 시킴

$$g_t = g_{t-1} + (\nabla f(x_{t-1}))^2$$

$$x_t = x_{t-1} - \frac{\eta}{\sqrt{g_t + \epsilon}} \cdot \nabla f(x_{t-1})$$

g_t: t번째 time step까지의 기울기 누적 크기

Feature 별로 학습률을 다르게 조절하는 것이 특징

Feature별 특성을 고려하여 학습을 효율적으로 돕는다는 장점이 있음

1) 큰 기울기를 가져 학습량이 많은 변수는 학습률을 감소

2)학습이 적게된 변수는 학습률 높여 조절

gt의 값은 점점 커져서 오래 진행하면 학습률이 0에 가까워져 더이상 학습이 진행되지 않을 수 있음
→학습이 잘 이루어져서 그런건지, 추가적으로 학습이 안 된건지 알기 어려움.

8.5.2 RMSProp

기울기 대신 지수 가중 이동 평균을 사용/ r = 지수이동평균의 업데이트 계수

$$g_t = \gamma g_{t-1} + (1-\gamma)(\nabla f(x_{t-1}))^2$$

$$x_t = x_{t-1} - \frac{\eta}{\sqrt{g_t + \epsilon}} \cdot \nabla f(x_{t-1})$$

- 변수(feature)마다 적절한 학습률을 적용하여 효율적인 학습을 진행할 수 있다는 점
- AdaGrad보다 학습을 오래 할 수 있다는 점
  (r을 사용하여 gt가 무한정 커지는 것을 방지)

8.5.3 Adam

운동량 알고리즘과 RMSProp의 장점을 결합한 알고리즘

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla f(x_{t-1})$$

$$g_t = \beta_2 g_{t-1} + (1 - \beta_2)(\nabla f(x_{t-1}))^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \hat{g}_t = \frac{g_t}{1 - \beta_2^t}$$

$$x_t = x_{t-1} - \frac{\eta}{\sqrt{\hat{g}_t + \epsilon}} \cdot \hat{m}_t$$

- Adam은 운동량을 기울기의 적률의 추정값으로써 직접 도입한다는 것

8.6 근사 2차 방법들

2차 방법들은 최적화 개선을 위해 2차미분(이계도함수) 활용

2차 방법들을 심층 신경망의 훈련에 적용하는 문제를 논의

8.6.1 뉴턴법

테일러 급수 전개를 이용해서 비용함수를 전개하는데 기초한 최적화 방안

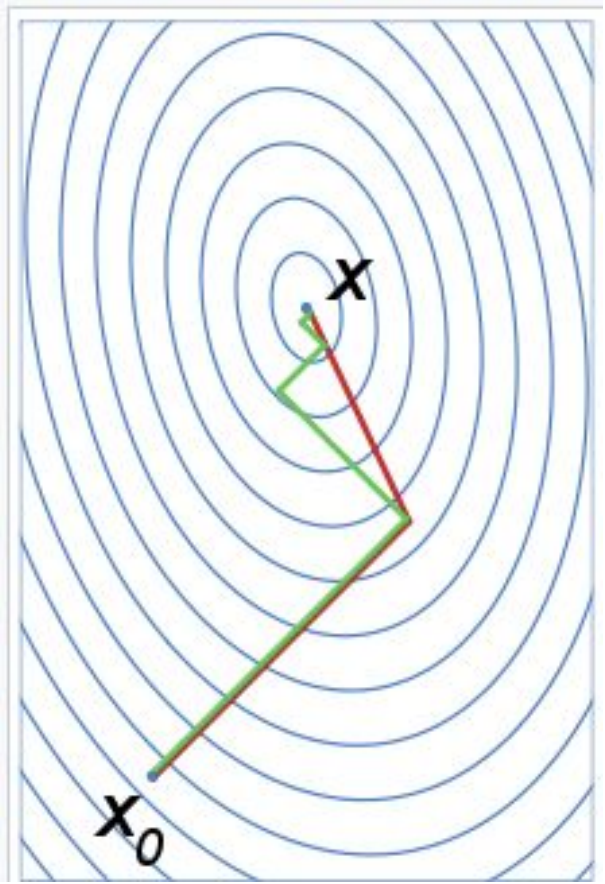
$$J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} L(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}), y^{(i)})$$

- 수렴 속도는 빠르지만 해에 수렴하지 않거나, 엉뚱한 해에 수렴 할 가능성 있음
- 도함수를 구하기 곤란한 경우 적용하기 어려움
- 계산 복잡도 O(k^3) 높음

8.6.2 켤레 기울기법

Ax=b를 풀로 싶을 때 A의 행렬이 너무 커지게 되면 b를 구하는 계산 자체가 부담

→ 직접 역행렬을 구하기 힘드니 켤레 기울기 법을 사용하여 근사해를 구함



8.6.3 BFGS

켤레 기울기법 처럼 계산 부담 없이 뉴턴법의 일부 장점을 취하려는 시도의 하나,
차이점은 뉴턴법의 갱신 규칙을 좀 더 직접적으로 근사한다는 것

메모리 제한-BFGS:

Hessian 행렬을 저장하는 대신 n차원의 벡터 몇 개만을 유지하여 *Hessian 행렬을 추정

*Hessian 행렬: 어떤 함수의 이계도함수를 행렬로 표현한 것

8.7 최적화 전략과 메타알고리즘

구체적인 알고리즘을 만들어 내거나

기존 알고리즘에 끼워 넣을 수 있는 서브루틴을 만들어 낼 수 있는

일반적인 틀에 해당하는 것들이 많음

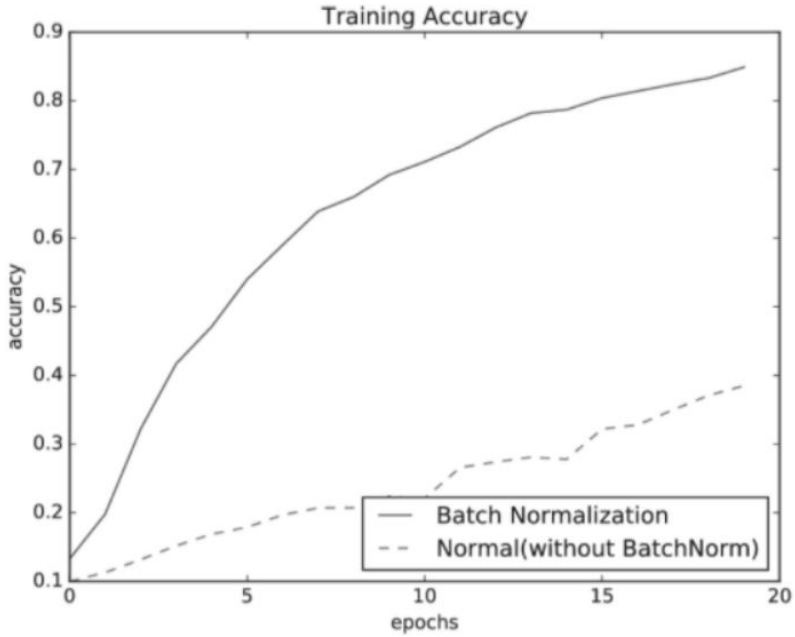
이번 절에서는 그런 일반적인 최적화 전략들을 살펴 볼 예정

8.7.1 배치정규화

깊은 모형의 훈련에서 발생하는 어려움을 극복하기 위해 만들어진 것

학습 과정에서 각 배치 단위 별로 데이터가 다양한 분포를 가지더라도 각 배치별로 평균과 분산을 이용해 정규화하는 것

- 학습을 빨리 진행 할 수 있음
- 초기값에 크게 의존 안 함
- 오버피팅 억제



8.7.2 좌표 하강법

모든 변수에 대해 한 번에 한 좌표씩 최적화 함

- 매우 간단하여 구현하기 쉬움
- 적합한 문제에 대해 주의깊게 구현될 경우 아주 좋은 성능을 보임
- 주어진 최적화 문제를 더 작은 조각들로 분해해서 빠르게 풀 수 있음

8.7.3 폴랴크 평균법

최적화 알고리즘이 매개변수 공간을 거쳐 간 궤적에 있는 여러 점의 평균을 구함

8.7.4 지도 사전 훈련

모델이 복잡하고 최적화하기 어렵거나 과제가 아주 어려울때,

더 간단한 모델을 훈련해서 과제를 푼 다음 그 모델을 좀 더 복잡하게 만들거나 더 간단한 과제를 풀도록 모델을 훈련한 수 그 모델을 최종 과제에 적용

8.7.5 최적화를 위한 모형 설계

최적화 알고리즘을 개선하는 것이 최고의 최적화 개선 전략은 아님

알고리즘을 바꾸는 대신 처음부터 최적화가 쉽도록 모델을 설계해서 최적화 개선하는 경우도 많음

강력한 최적화 알고리즘을 사용하는 것보다 최적화하기 쉬운 모델을 선택하는 것이 더 중요

8.7.6 연속법과 커리큘럼 학습

최적화의 여러 어려움은 비용함수의 전역 구조에서 비롯

단지 추정값을 개선하거나 국소(local) 갱신 방향을 개선한다고 해서 그런 어려움이 개선되지 않음

연속법은 국소 최적화 과정이 그러한 바람직한 영역에서 대부분의 시간을 보내도록 초기점들을 선택함으로써 최적화를 쉽게 만드는 일단의 전략들을 통칭

연속법의 핵심은 같은 매개변수에 관하 일련의 목적 함수들을 구축 하는 것

<커리큘럼 학습>

- 커리큘럼 학습은 일반적으로 사람이 “초급 수준의 학습부터 대학 수준의 학습내용까지” 긴 기간을 가지고 학습하는 경우를 의미하는데, 이를 머신러닝의 학습에 적용해보자는 것이다.
- 커리큘럼 학습은 일반화와 빠른 수렴 속도의 장점을 가진다

감사합니다