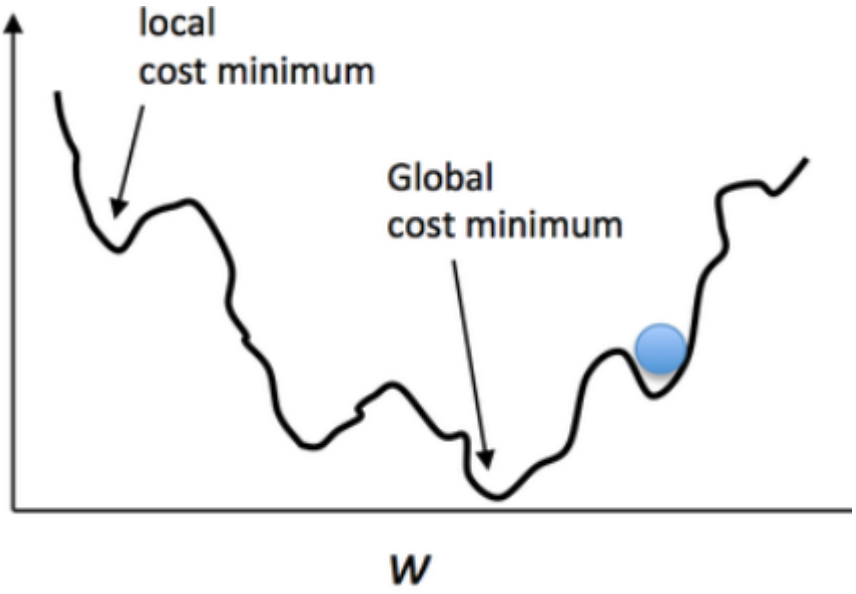
Local Minima와 Global Minima

생성일	@2022년 3월 3일 오후 4:32
속성	머신러닝
태그	면접질문

Local Minima와 Global Minima에 대해 설명해주세요.

- Gradient Descent(경사하강법)을 통해 Cost Function의 최소값을 찾게 되도록 하는데 *극소점 즉, Global Minima에 다다르지 않고, 기울기가 0이 되는 다른 지점을 Local Minima*라고 합니다.
- 이렇게 Local Minima로 빠지는 것을 방지하기 위한 방법으로 다양한 Optimize 기법이 있습니다.



- 이걸 방지하기 위해서는 w의 범위를 적절하게 부여하거나, 초기값(initial value)을 훌륭한 값으로 주어줘야 한다

gradient descent Oscillation

- gradient descent에서 학습률이 낮으면 천천히 가는 대신 최솟값 근처에서 안정적으로 수렴할 수 있지만, 학습률이 높으면 빠르게 가는 대신 최솟값 근처에서 왔다 갔다 할 수

있다고 하였습니다.

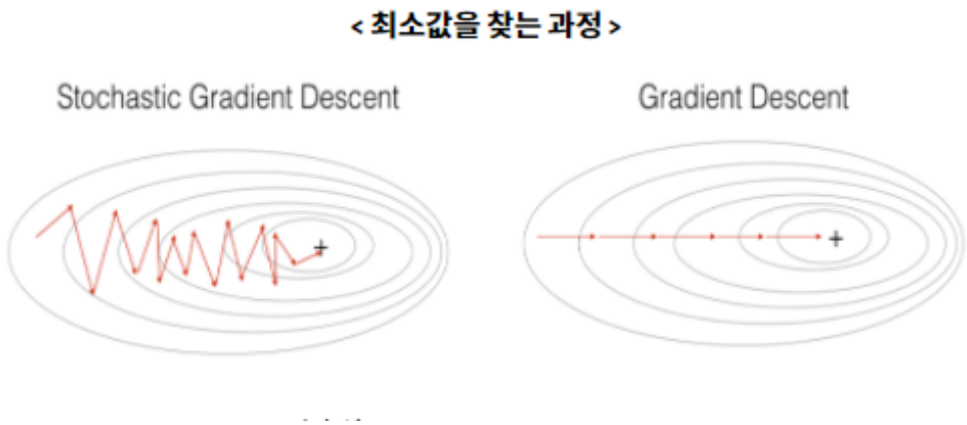
- 이 *왔다 갔다 하면서 최솟값을 찾는데 시간을 낭비하는 현상을 Oscillation 문제*라고 합니다.

Optimizer 중에 아는게 있나요?

- *Stocatic Gradient Descent 방법은 데이터의 전체를 활용하지 않고 일부분을 활용하면서 업데이트 하는 방식으로 Local Minima에 빠지더라도 다른 데이터 셋이 사용 되었을 때, 빠져나올 수 있다는 장점을 가지고 있습니다*. 기존에는 스무스하게 찾아갔다면, 이는 지그재그 형식으로 Global Minima를 찾아가게 됩니다.
- 그 외에도 방향성을 유지하는 *Momentom 기법이나, 업데이트하는 사이즈를 조정하는 Adagrad방식, 이 둘을 적절히 합친 Adam 기법*이 있는 것으로 알고 있습니다.

stochastic gradient descent

- 전체 데이터를 사용하는 것이 아니라, *랜덤하게 추출한 일부 데이터를 사용*하는 것이다. 따라서 *학습 중간 과정에서 결과의 진폭이 크고 불안정하며, 속도가 매우 빠르다*. 또한, 데이터 하나씩 처리하기 때문에 오차율이 크고 *GPU의 성능을 모두 활용하지 못하는 단점*을 가진다. 이러한 단점들을 보완하기 위해 나온 방법들이 ***Mini batch***를 이용한 방법이며, 확률적 경사 하강법의 노이즈를 줄이면서도 전체 배치보다 더 효율적인 것으로 알려져 있다.



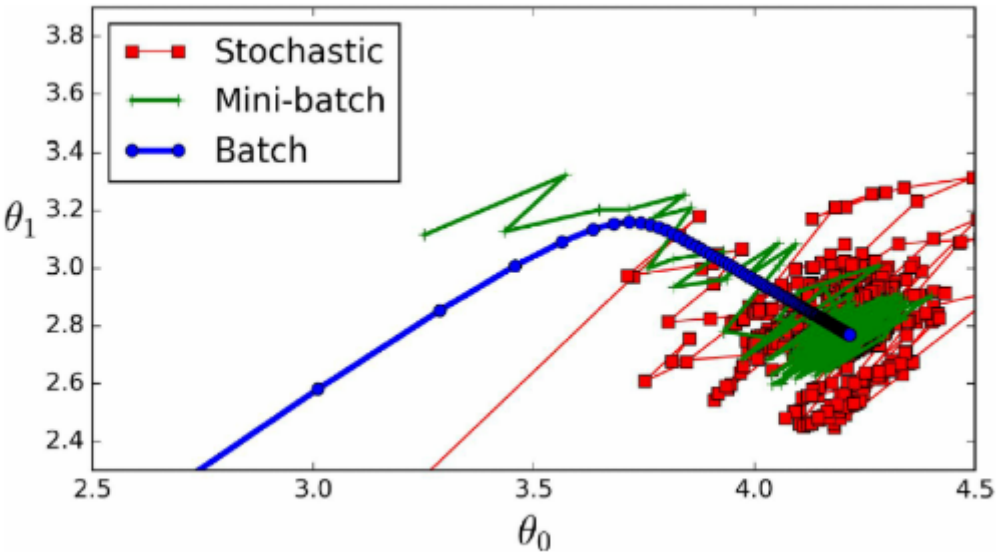


Figure 4-11. Gradient Descent paths in parameter space

- **Batch**는 *모든 데이터를 한꺼번에 학습*하는데 활용하기 때문에 부드럽게 수렴하나 샘플의 개수만큼 계산해야 하기 때문에 시간이 다소 소요된다.
- **Stochastic** 방법은 데이터를 *한개씩 추출해서 처리*해보고 이를 모든 학습 데이터에 대해 적용한 것이다.
- **Mini-Batch** 는 전체 학습 데이터를 배치 사이즈로 나누어서 순차적으로 진행한다. 일반적으로 딥러닝 학습에 사용되는 방법이며, *Batch 보다 빠르고 SGD 보다 낮은 오차율을 가진다*.

momentum

- 기본적으로 GD와 동일하게 경사를 하강하는 알고리즘.
- 경사를 하강할 때, 바로 이전의 갱신량을 참고해서 새로운 갱신량에 추가해줌으로, 일종의 가속도를 구현한 것입니다
- **lr이 너무 크면 갱신값이 이리저리 튀어서 최소점을 찾기 어렵고, lr이 너무 작으면, 지역 최소점에 갇히기 쉽죠.** 이것을 해결하기 위해서는, **최적점에서 멀리 있을 때는 lr을 크게 하고, 점차 최소점에 나아갈수록 lr을 작게 만드는 것이 좋을 것**입니다. momentum이 바로 이러한 문제 해결을 위해 만들어진 옵티마이저이고,
- 그냥 편하게 GD의 가중치 갱신량에 운동량 계수 momentum에 따라 정해진 가속도를 더해준다고 생각하시면 됩니다.
- Adagrad를 사용하면 학습을 진행하면서 굳이 step size decay등을 신경써주지 않아도 된다는 장점이 있다. 보통 adagrad에서 step size로는 0.01 정도를 사용한 뒤, 그 이후

로는 바꾸지 않는다. 반면, Adagrad에는 학습을 계속 진행하면 step size가 너무 줄어든다는 문제점이 있다

- 학습이 오래 진행될 경우 step size가 너무 작아져서 결국 거의 움직이지 않게 된다. 이를 보완하여 고친 알고리즘이 RMSProp과 AdaDelta이다.

RMSProp

- RMSProp은 딥러닝의 대가 제프리 힌톤이 제안한 방법으로서, Adagrad의 단점을 해결하기 위한 방법이다. Adagrad의 식에서 gradient의 제곱값을 더해나가면서 구한 GtGt 부분을 합이 아니라 지수평균으로 바꾸어서 대체한 방법이다.

AdaGrad

- Adagrad(Adaptive Gradient)는 변수들을 update할 때 각각의 변수마다 step size를 다르게 설정해서 이동하는 방식이다. 이 알고리즘의 기본적인 아이디어는 '지금까지 많이 변화하지 않은 변수들은 step size를 크게 하고, 지금까지 많이 변화했던 변수들은 step size를 작게 하자' 라는 것이다.
- 자주 등장하거나 변화를 많이 한 변수들의 경우 optimum에 가까이 있을 확률이 높기 때문에 작은 크기로 이동하면서 세밀한 값을 조정하고, 적게 변화한 변수들은 optimum 값에 도달하기 위해서는 많이 이동해야할 확률이 높기 때문에 먼저 빠르게 loss 값을 줄이는 방향으로 이동하려는 방식이라고 생각할 수 있겠다.

Adam

현재 가장 자주 사용되는 옵티마이저입니다.

옵티마이저를 모른다면 일단 이걸 사용해도 될 정도입니다. (많은 사용을 통해 검증된 알고리즘이라고 하죠.)

여러모로 이전 옵티마이저들의 장점을 취해 만들어졌다고 하더군요.

(combination of RMSprop and Stochastic Gradient Descent with momentum)

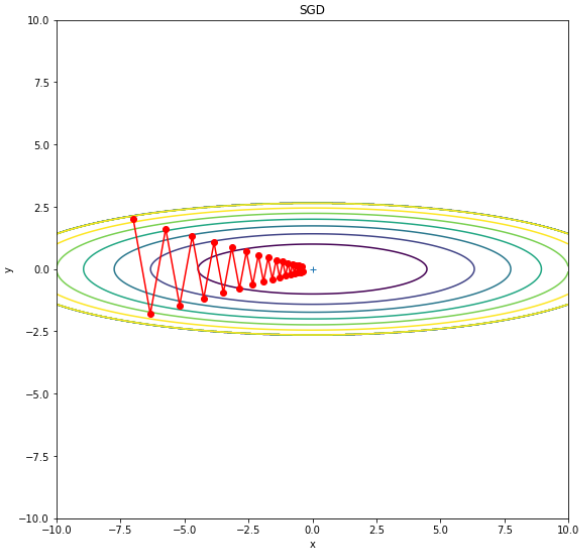
이름의 의미는,

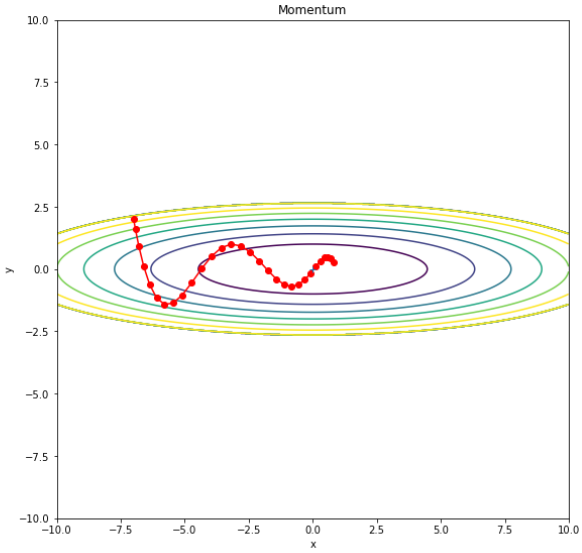
Adaptive momentum estimation입니다.

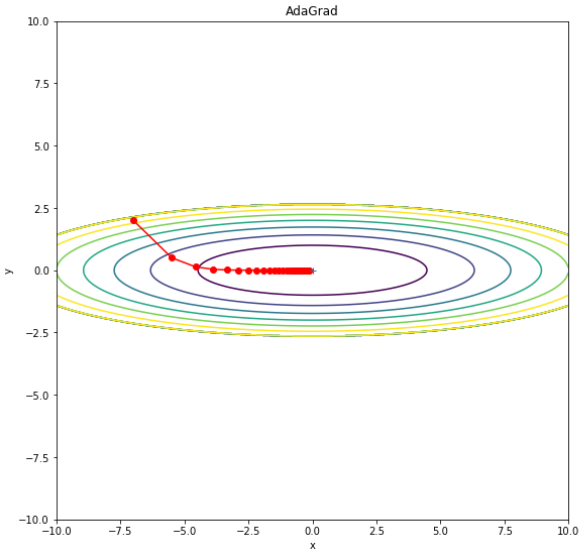
이름 그대로 Adagrad와 momentum의 결합으로,

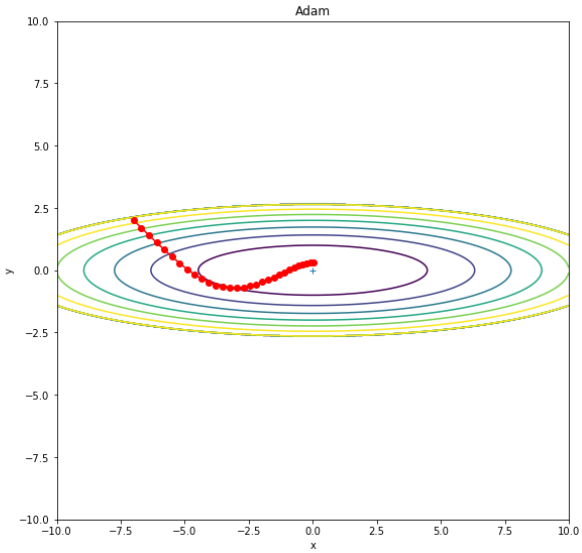
이전까지는 momentum의 성능이 매우 뛰어났기에, 상황에 따라 다른 옵티마이저를 사용하면서도, momentum을 주로 사용하였는데,

Adam이 나온 이후로는, 보다 폭넓은 딥러닝 아키텍쳐에서 좋은 성능을 보이는 이 옵티마이저가 주목받고 검증되었다고 합니다.









1. 발산 가능성은?
2. SDG는 정확히 어떻게 진행되는가