Gradient Descent (2)

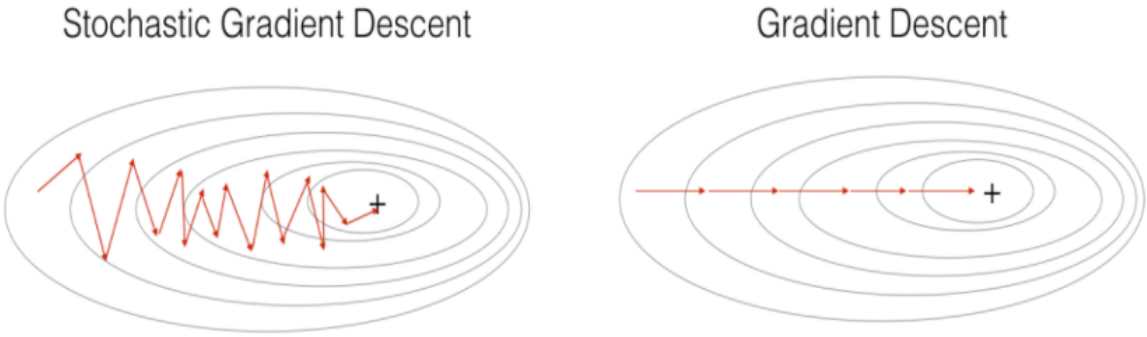
Finite Difference Method (유한차분법)

: 함숫값의 차를 이용해 편미분방정식 푸는 방법. 테일러 급수 전개로 유도

SGD (Stochastic Gradient Descent; 확률적 경사 하강법)

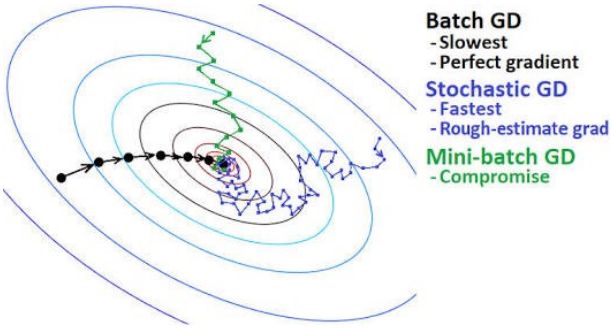
$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial L}{\partial \mathbf{W}}$$

- w=갱신할 가중치 매개변수, dL/dW=w에 대한 손실함수의 기울기



: 랜덤하게 추출한 일부 데이터에 대해서 가중치 업데이트 시 한 개의 데이터를 이용해서 그래디언트 계산하고 GD 알고리즘 적용하는 방법

- 장점: 속도 빠름, 메모리 소비 낮음
- 단점: 학습 중 결과의 진폭이 크고 불안정하며 (분산이 커짐) 데이터를 하나씩 처리하기 때문에 오차율 큼 → mini batch 활용해서 보완



SGD 단점 개선: momentum, AdaGrad, Adam

Optimization

SGD 이외의 최적화 함수들

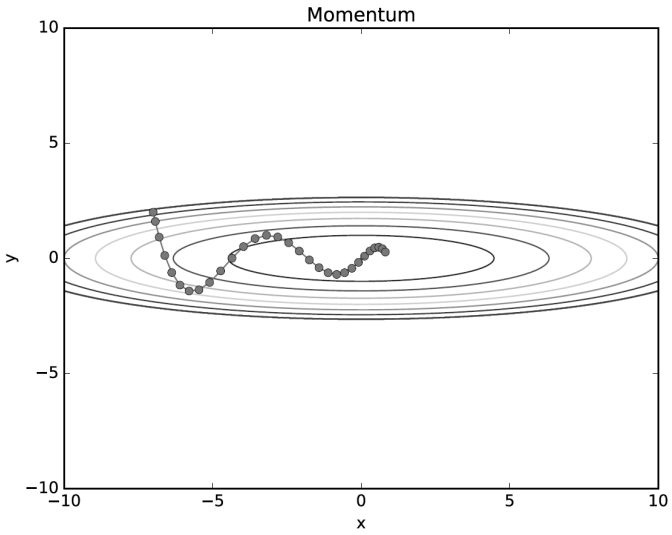
Momentum

: "운동량"을 뜻하는 단어. 물리와 관계 있음

$$\mathbf{v} \leftarrow \alpha\mathbf{v} - \eta \frac{\partial L}{\partial \mathbf{W}}$$

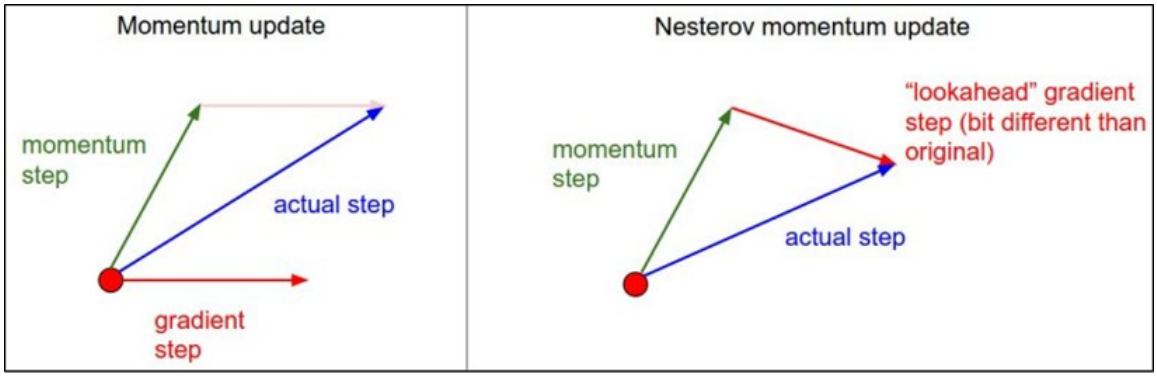
$$\mathbf{W} \leftarrow \mathbf{W} + \mathbf{v}$$

- η=학습률, v=속도, αv는 물체가 아무런 힘을 받지 않을 때 서서히 하강시키는 역할. (α=0.9)
  - 첫번째 식은 물체가 기울기 방향으로 힘을 받아 가속된다는 물리 법칙을 나타냄
  - 두번째 식에 의해서 속도만큼 위치 (매개변수)가 이동함



NAG (Nesterov Accelerated Gradient; 네스테로프 모멘텀)

: 모멘텀의 방향으로 조금 앞선 위치에서 그래디언트 계산함



Difference between Momentum and NAG. Picture from CS231.

- 이동될 방향을 미리 예측함으로써 불필요한 이동을 줄임

Adagrad

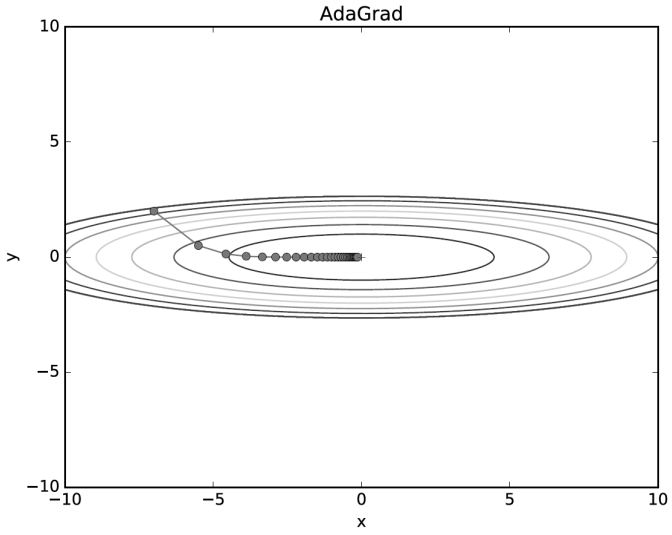
: 학습률을 서서히 낮추기 위해 (learning rate decay) 각각의 매개변수에 "맞춤형" 값을 만들어주는 방법

$$\mathbf{h} \leftarrow \mathbf{h} + \frac{\partial L}{\partial \mathbf{W}} \odot \frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{1}{\sqrt{\mathbf{h}}} \frac{\partial L}{\partial \mathbf{W}}$$

- h = 기존 기울기 값을 제곱하여 계속 더해주는 변수. 매개변수 업데이트 시 1/sqrt(h)를 곱해서 학습률 조정

- 이렇게 하면 학습률 감소가 매개변수의 원소마다 다르게 적용되고,과거의 기울기를 제곱하여 계속 더해가기 때문에 학습이 진행될수록 갱신 강도가 약해진다



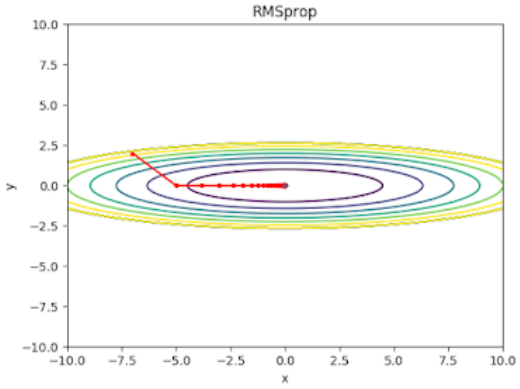
- y축 방향을 갱신 강도가 약해지고 지그재그 움직임이 줄어듦
- 단, 무한 학습시 갱신량이 0이 되어 전혀 갱신되지 않음. → 개선방법: RMSProp

RMSProp

: 과거의 모든 기울기를 똑같이 더해가는 것이 아닌, 먼 과거의 기울기는 서서히 잊고 새로운 기울기 정보를 크게 반영하여 매개변수 업데이트 하는 방법

$$h \leftarrow \gamma h + (1-\gamma)\frac{\partial L}{\partial W} \odot \frac{\partial L}{\partial W}$$
$$W \leftarrow W - \eta \frac{1}{\sqrt{h+\epsilon}} \frac{\partial L}{\partial W}$$

- γ=decay rate



- AdaGrad 갱신량이 0이 될 때의 문제점 개선하기 위한 알고리즘이다보니 큰 차이가 보이지 않음

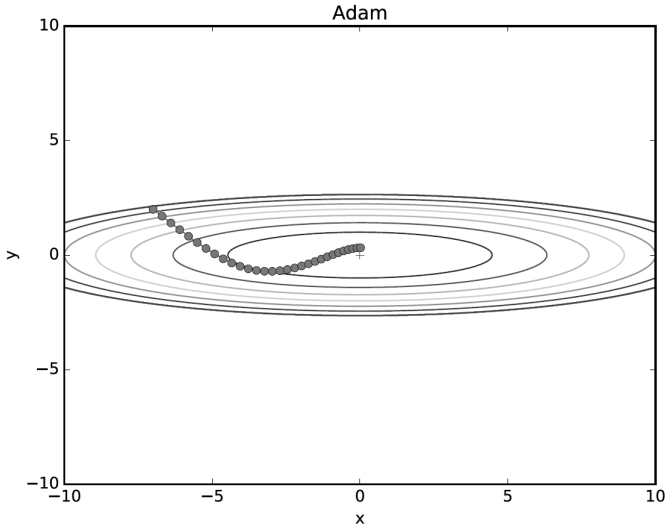
AdaDelta

: AdaGrad에서 발생한 매 iteration마다 학습률이 작아지는 문제 & 하이퍼파라미터 η가 필요한 문제 해결하고자 함

Adam

: 모멘텀의 움직임과 ADaGrad의 매개변수 원소마다 갱신 정도 조정하는 기법을 합친 방법

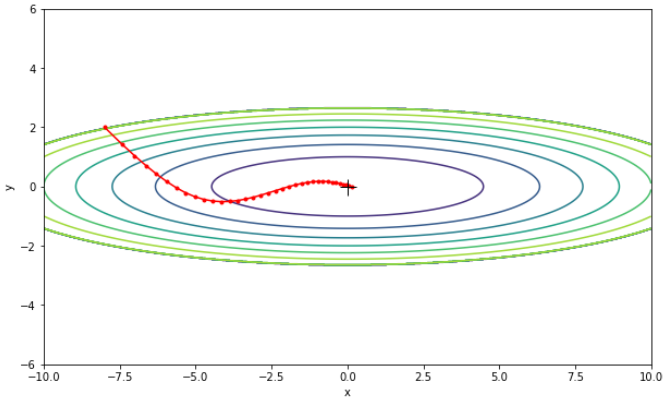
- 하이퍼파라미터를 총 세개 설정한다. 하나는 지금까지의 학습률이고 나머지는 일차 모멘텀용 계수와 이차 모멘텀용 계수임.



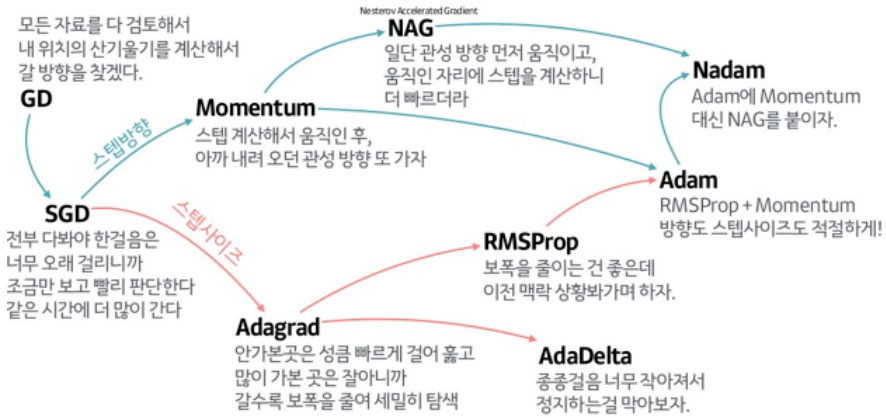
Nadam

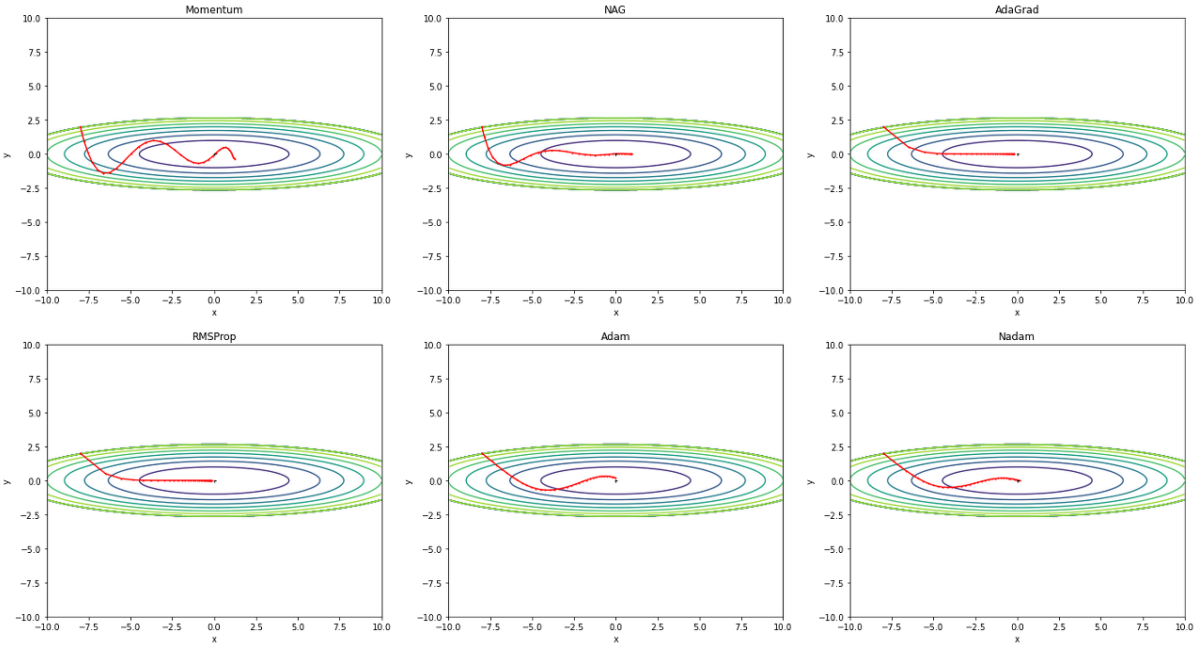
: Adam에서 적용한 모메텀 기법을 NAG로 변경한 방식

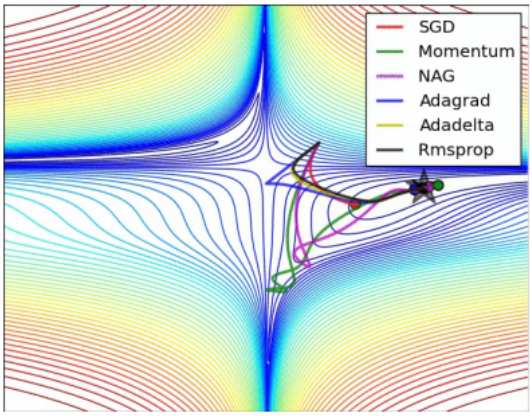
- Adam보다 더 빠르고 정확하게 전역 최솟값 찾을 수 있음



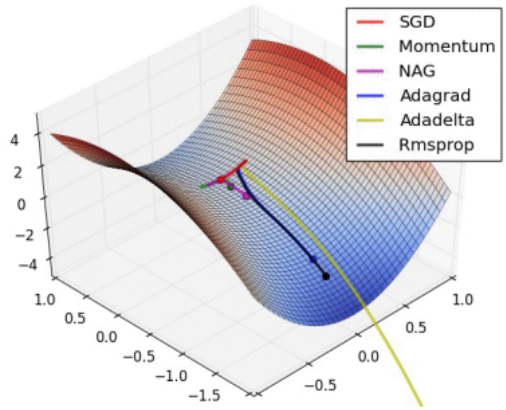
정리







(a) SGD optimization on loss surface contours



(b) SGD optimization on saddle point

Figure 4: Source and full animations: Alec Radford

애니메이션 링크

참고

https://james-scorebook.tistory.com/entry/옵티마이저Optimizer-22 (코드포함)

https://koreanfoodie.me/178

https://go-hard.tistory.com/11

https://twinw.tistory.com/247

http://bakbang.blogspot.com/2017/08/rmsprop.html

https://hiddenbeginner.github.io/deeplearning/2019/09/22/optimization_algorithms_in_deep_learning.html

https://www.slideshare.net/yongho/ss-79607172

"Gradient descent based optimization algorithms for deep learning models training." *arXiv preprint arXiv:1903.03614* (2019).

Ruder, Sebastian. "An overview of gradient descent optimization algorithms." *arXiv preprint arXiv:1609.04747* (2016).