[Week10]_문가을

딥러닝 2단계 : 심층 신경망 성능 향상 시키기

4. 최적화 알고리즘

미니 배치 경사 하강법

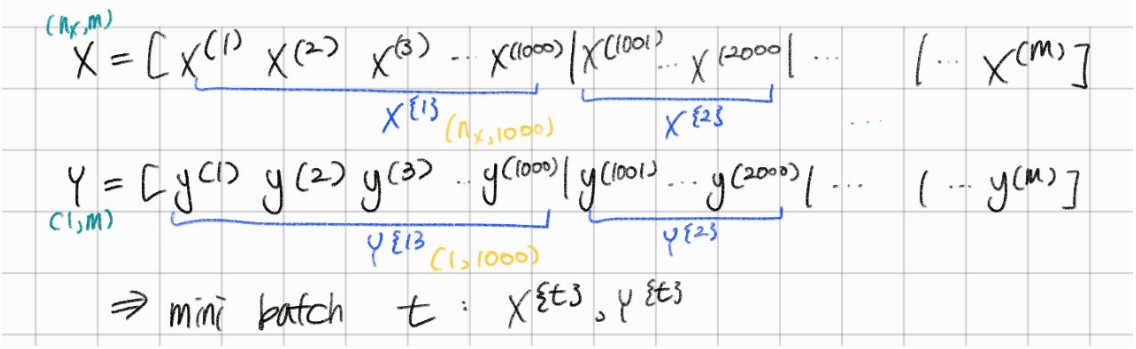
작은 훈련 세트 → 미니 배치

큰 데이터 세트에서 훈련하는 것은 매우 느린 과정임.

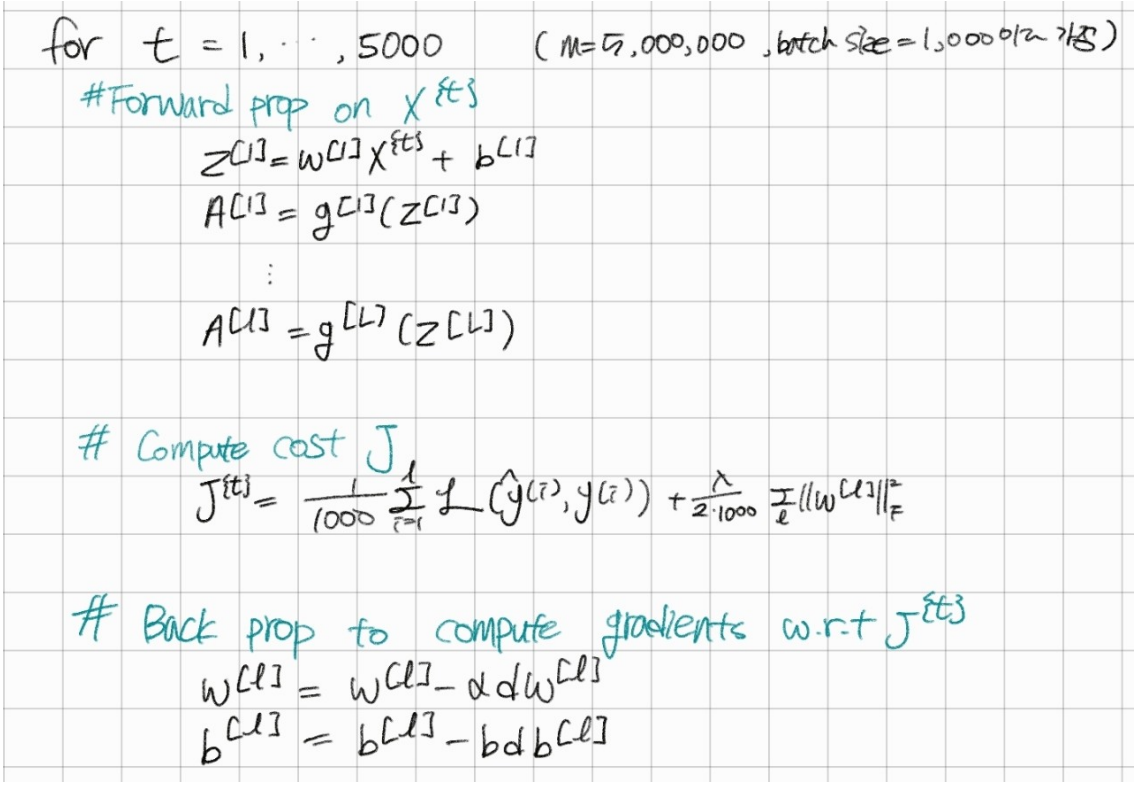
→ 경사하강법의 한 단계를 할 때마다 전체 훈련 샘플을 처리해야 하기 때문임.

- 벡터화는 m개 샘플에 대한 계산을 효율적으로 만들어줌

벡터화한 샘플에서 미니 배치로 쪼갬.

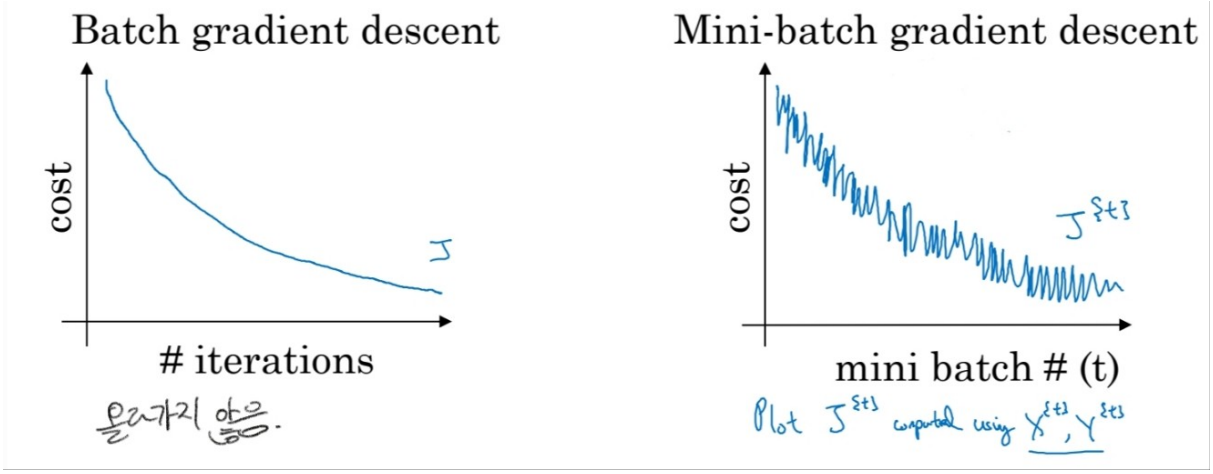


- 미니 배치 경사 하강법 구현



에포크 : 훈련 세트를 거치는 한 반복

미니 배치 경사하강법 이해하기



노이즈가 발생하는 이유 :

각 미니 배치마다 학습하기 좋은 데이터냐에 따라 비용이 차이가 있음.

<미니 배치 사이즈 결정>

1) stochastic : 하나의 샘플만 처리하여 간단함. 노이즈도 작은 학습률을 사용해 해결할 수 있음.

단점 : 벡터화에서 얻을 수 있는 속도 향상을 잃게 됨.

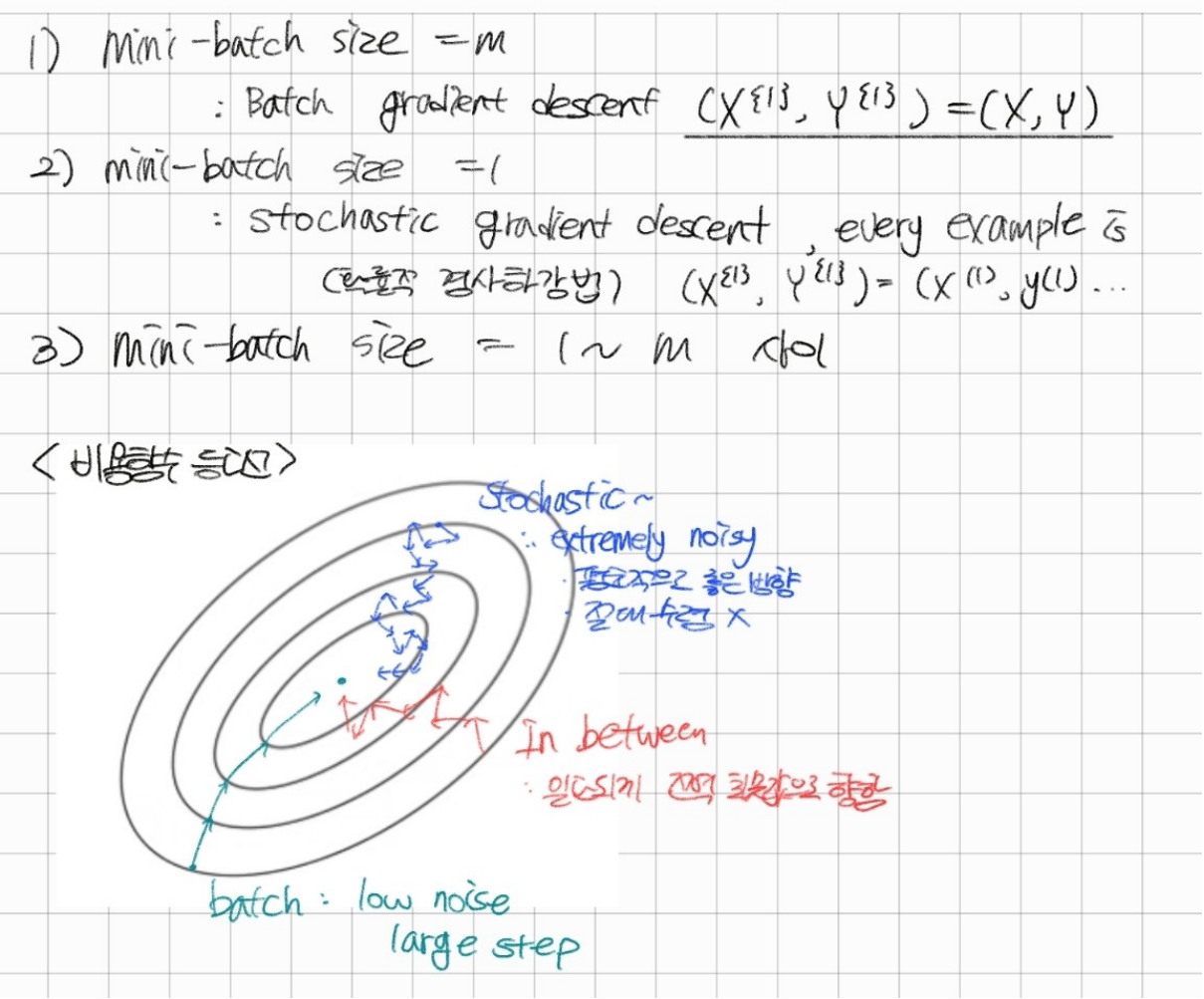
2) In between : 가장 빠른 학습 제공

장점 :

- 많은 벡터화를 얻음. → 한번에 샘플을 처리하는 속도가 빠름
- 전체 훈련 세트가 진행되기를 기다리지 않고, 진행할 수 있음.

3) batch : 매우 큰 훈련 세트를 모든 반복에서 진행함.

단점 : 한 반복에서 너무 오랜 시간이 걸림.



<미니 배치 사이즈 고르는 가이드라인>

- 작은 훈련 세트라면(샘플이 2000개 이하) → 배치 경사 하강법
- typical mini-batch size : 2의 제곱 → 64~512 (일반적)

모든 X^{t}와 Y^{t}가 CPU, GPU 메모리에 맞는지 확인하기

→ 맞지 않으면 성능이 훨씬 나빠짐.

지수 가중 이동 평균 (Exponentially Weighted Average)

경사 하강법 보다 빠른 최적화 알고리즘

- 최근의 데이터에 더 많은 영향을 받는 데이터들의 평균 흐름을 계산하기 위해 지수 가중 이동 평균을 구한다. 지수 가중 이동 평균은 최근 데이터 지점에 더 높은 가중치를 준다.
- θt 를 t 번째 날의 기온이라고 했을 때, 지수 가중 이동 평균(vt)의 식은 다음과 같다.

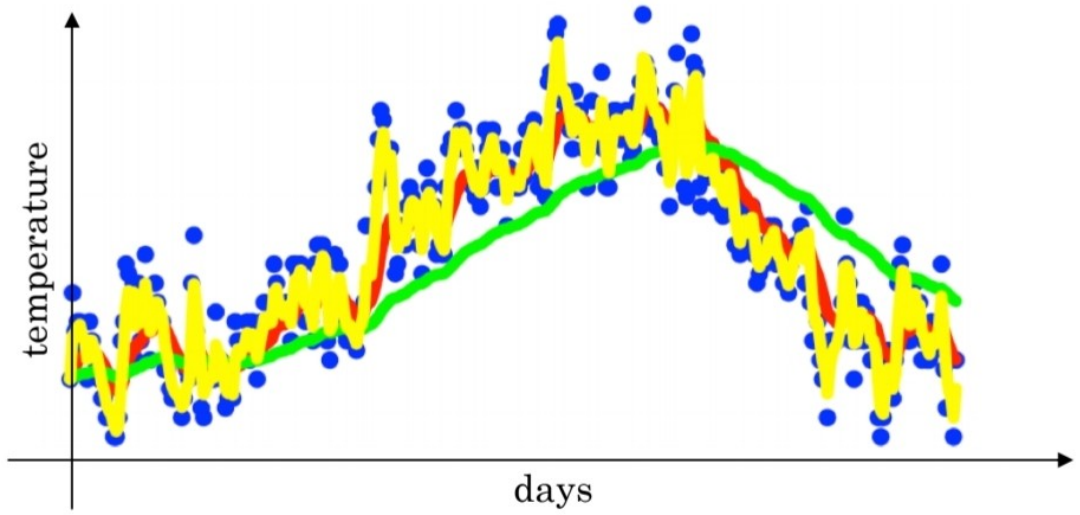
$$v_t = \beta v_{t-1} + (1-\beta)\theta_t$$

- 이 때 β 값은 하이퍼 파라미터로 최적의 값을 찾아야 하는데, 보통 사용하는 값은 0.9 임.
- vt 는 $1-\beta 1$ 기간 동안 기온의 평균을 의미함.

- β = 0.9일 때 10일의 기온 평균

- β = 0.5일 때 2일의 기온 평균 (더 노이즈가 많고, 이상치에 민감함/ 기온 변화에 빠르게 적응함)

지수 가중 이동 평균 이해하기



beta = 0.9 : 빨간색 곡선

beta = 0.98 : 초록곡선

beta = 0.5 : 노란색 곡선

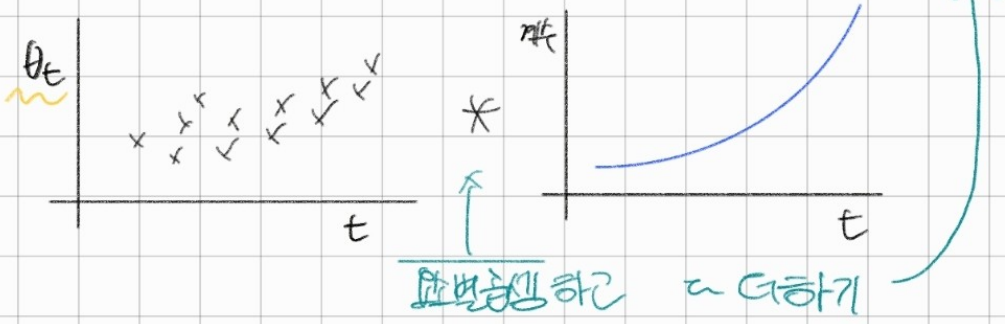
$$v_t = \beta v_{t-1} + (1-\beta)\theta_t$$

$$v_{100} = 0.9v_{99} + 0.1\theta_{100}$$
$$v_{99} = 0.9v_{98} + 0.1\theta_{99}$$
$$v_{98} = 0.9v_{97} + 0.1\theta_{98}$$

$v_{100} = 0.1\theta_{100} + 0.9v_{99}$ $\quad 0.1\theta_{98} + 0.9v_{97}$

$(0.1\theta_{99} + 0.9v_{98})$

$= 0.1\theta_{100} + 0.1 \times 0.9 \times \theta_{99} + 0.1 \times (0.9)^2 \times \theta_{98} + 0.1 \times (0.9)^3 \theta_{97} + \cdots$



- 얼마의 기간이 이동하면서 평균이 구해졌는가? 는 아래의 식으로 대략적으로 구할 수 있습니다.
  - $\beta = (1-\varepsilon)$ 라고 정의 하면
  - $(1-\varepsilon)^n = \frac{1}{e}$ 를 만족하는 n 이 그 기간이 되는데, 보통 $\frac{1}{\varepsilon}$ 으로 구할 수 있습니다.

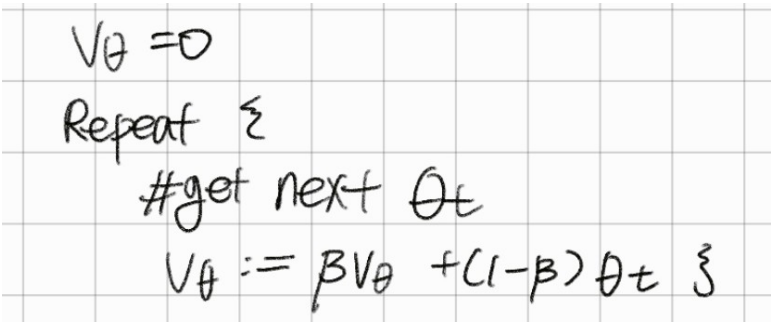
$(0.9)^{10} \approx 0.35 \approx \frac{1}{e}$ $\qquad (1-\varepsilon)^{\frac{1}{\varepsilon}} = \frac{1}{e}$

10일 뒤에는 가중치가 현재 날짜의 가중치의 1/3으로 줄어든다.

$(0.98)^{50} \approx \frac{1}{e}$

처음 50일 동안의 1/e 보다 가충치는 더 커질 것임.

<지수 가중 이동 평균 구현>

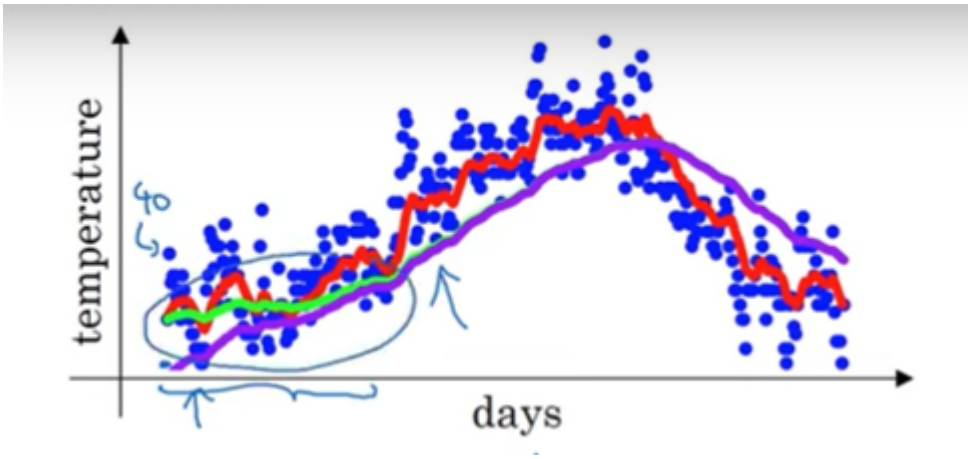


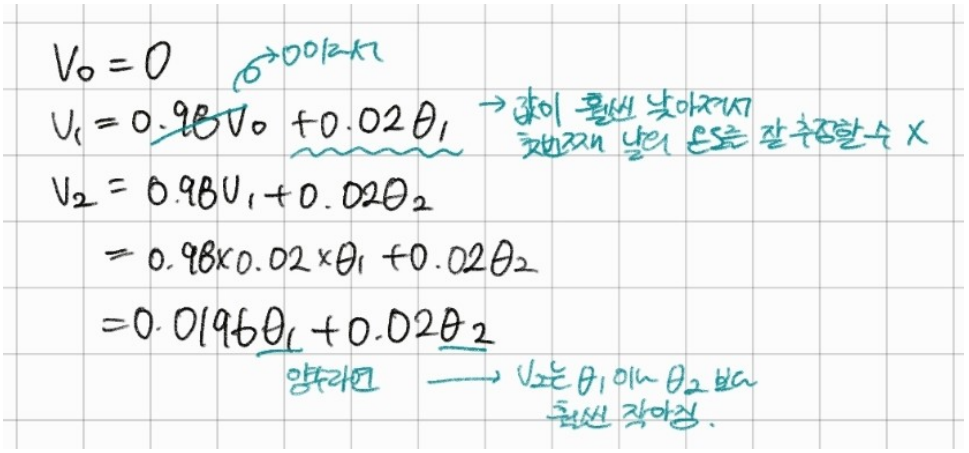
지수 평균을 얻는 식의 장점

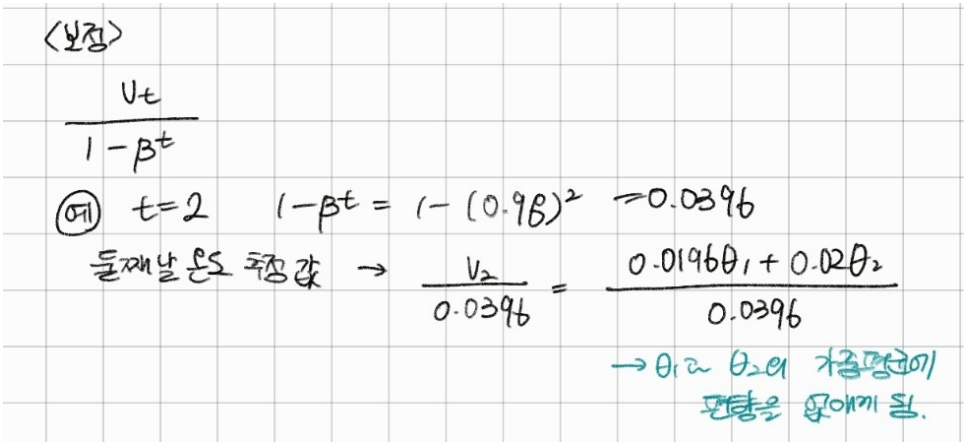
- 아주 적은 메모리 사용 : 초기화한 실수 값에 가장 최근에 얻은 값을 덮어쓰기만 하면 되기 때문임.
- 한 줄의 코드만 작성하면 되기 때문에 효율적임.

지수 가중 이동 평균의 편향 보정 (bias correction)

초기 단계 학습에서 더 나은 추정값을 얻도록 함.







t 가 더 커질수록 beta^t는 0에 가까워짐.

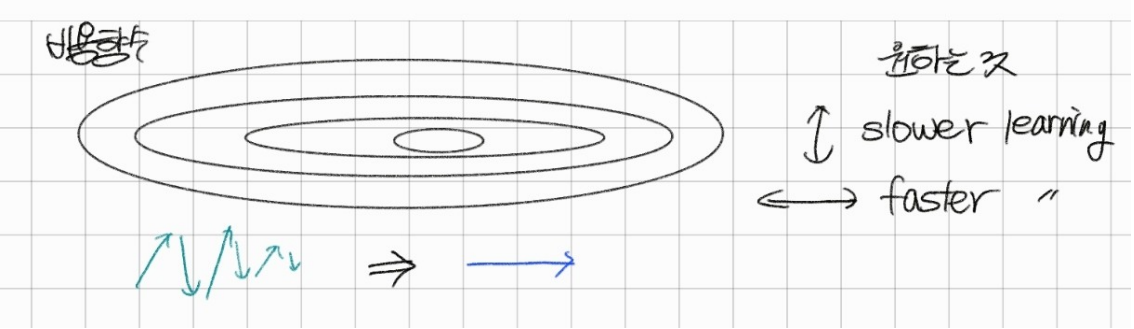
→ t가 충분히 커지면 편향 보정은 그 효과가 거의 없어짐.

머신러닝에서 지수가중평균을 구현할 때 대부분은 편향 보정을 거의 구현하지 않음 → 초기 단계를 기다리고 편향된 추정이 지나간 후부터 시작하기 때문임.

초기 단계의 편향이 신경 쓰인다면 편향 보정 사용.

Momentum 최적화 알고리즘

경사에 대한 지수가중평균을 계산하는 것. 그 가중치로 업뎃함.

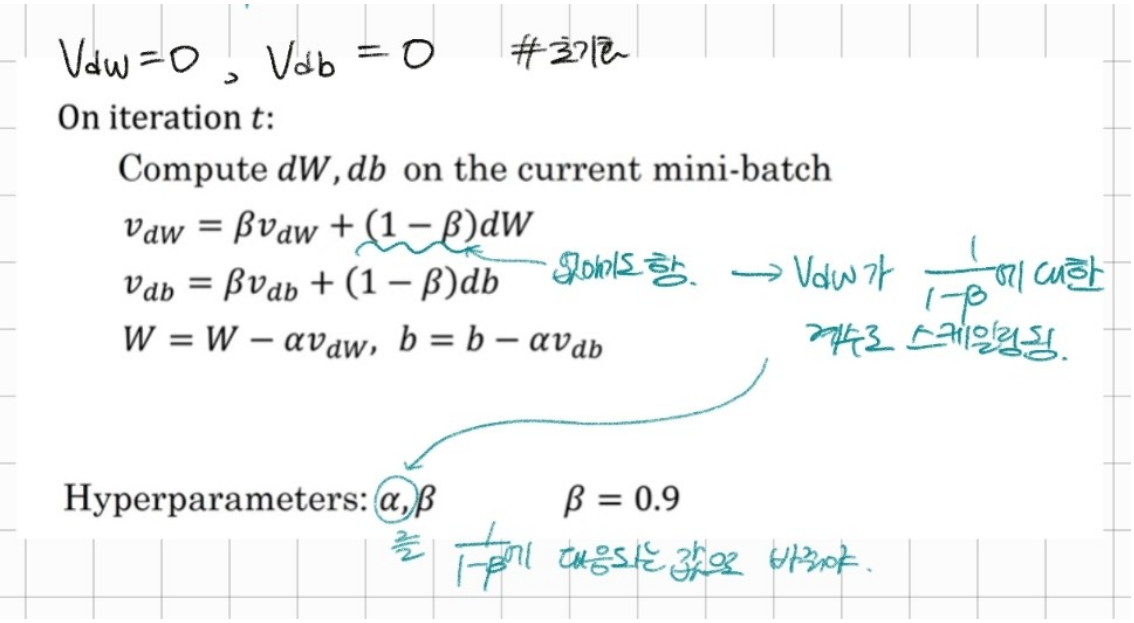


경사 하강법의 단계를 부드럽게 만들어줌.

→ 평균을 취하기 때문에, 수직방향의 진동이 0에 가까운 값으로 평균이 만들어짐. / 수평 방향에서는 모두 오른쪽을 향하고 있기 때문에, 평균은 꽤 큰 값을 가짐.

→ 수직 방향에서는 훨씬 더 작은 진동 / 수평 방향에서는 더 빠르게 움직이게 함.

<구현 방법>

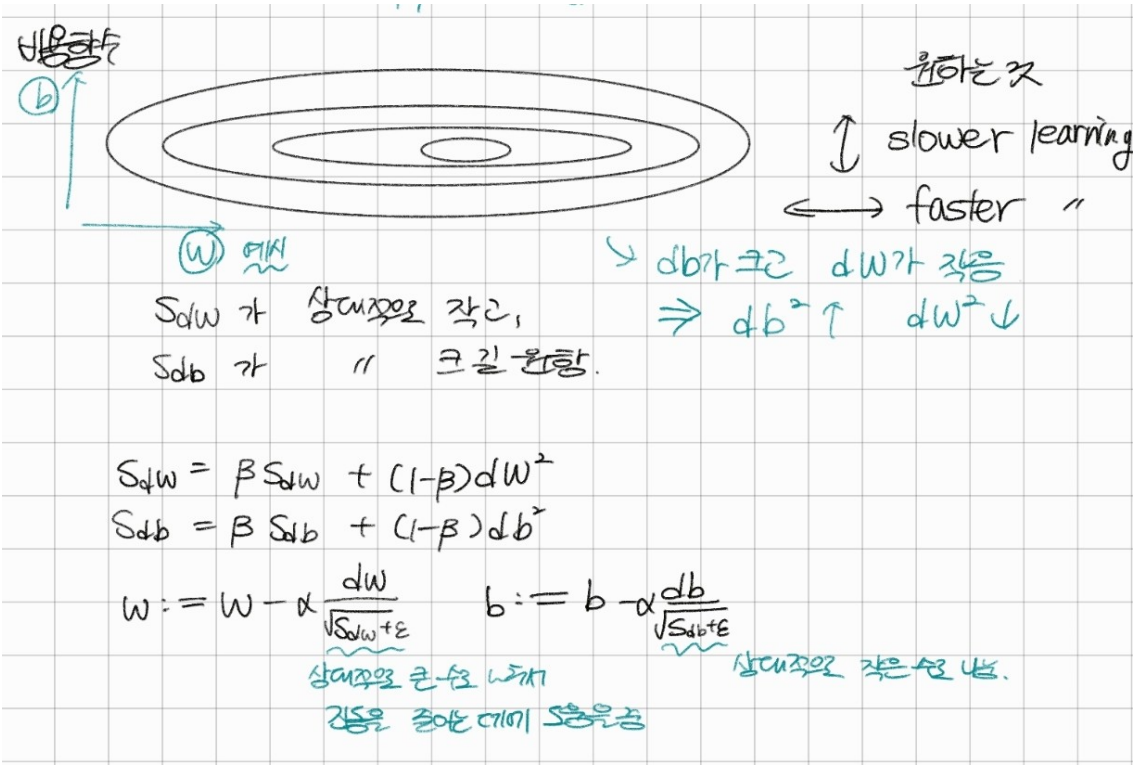


RMSProp 최적화 알고리즘

root mean square prop

- $S_{dW} = \beta_2 S_{dW} + (1-\beta_2) dW^2$
- 업데이트: $w := w - \alpha \frac{dW}{\sqrt{S_{dW} + \epsilon}}$
- dW^2 은 요소별 제곱을 뜻합니다.

도함수의 제곱을 지수가중평균함.



엡실론은 분모가 0에 매우 가까워, w가 폭발하지 않기 위해 아주 작은 값으로 설정함.

효과

- 큰 학습률을 사용해 빠르게 학습하고, 발산하지 않음.

Adam 최적화 알고리즘

Momentum 과 RMSProp 을 섞은 알고리즘.

- $V_{dW} = 0, S_{dW} = 0$ 로 초기화 시킵니다.
- Momentum 항: $V_{dW} = \beta_1 V_{dW} + (1 - \beta_1) dW$
- RMSProp 항: $S_{dW} = \beta_2 S_{dW} + (1 - \beta_2) dW^2$
- Bias correction: $V_{dW}^{correct} = \frac{V_{dW}}{1 - \beta_1^t}, S_{dW}^{correct} = \frac{S_{dW}}{1 - \beta_2^t}$
- 업데이트: $w := w - \alpha \frac{V_{dW}^{correct}}{\sqrt{S_{dW}^{correct}} + \epsilon}$

α : 매우 중요하고 보정될 필요가 있으므로 다양한 값을 시도해야 함.

β1 : 보통 0.9로 선택함.

β2 : 0.999

ϵ : 10^(-8) 값이 크게 중요하지는 않음.

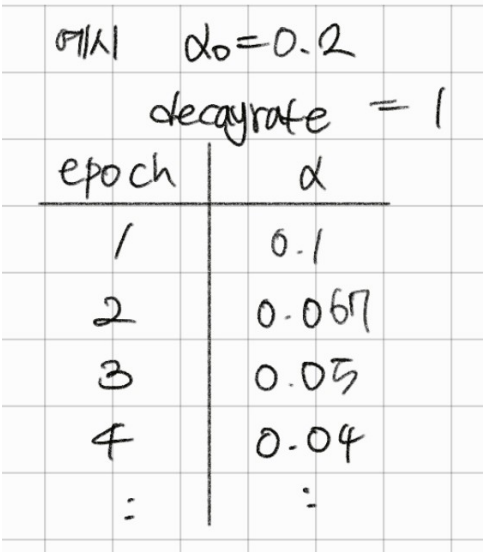
Adam : Adaptive moment estimation

학습률 감쇠 (learning rate decay)

시간에 따라 학습률을 천천히 줄여 학습 알고리즘의 속도를 높임.

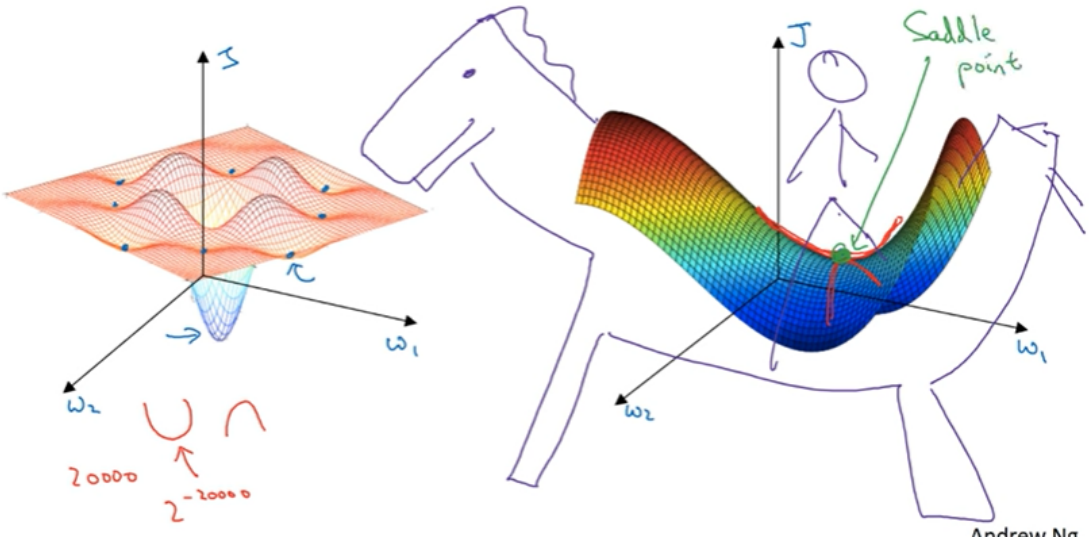
→ 학습이 수렴할수록 작은 스텝으로 진행

- 1 epoch = 전체 데이터를 1번 훑고 지나가는 횟수 입니다.
- $\alpha = \frac{1}{1 + decay\ rate\ \times epoch\ num} \alpha_0$

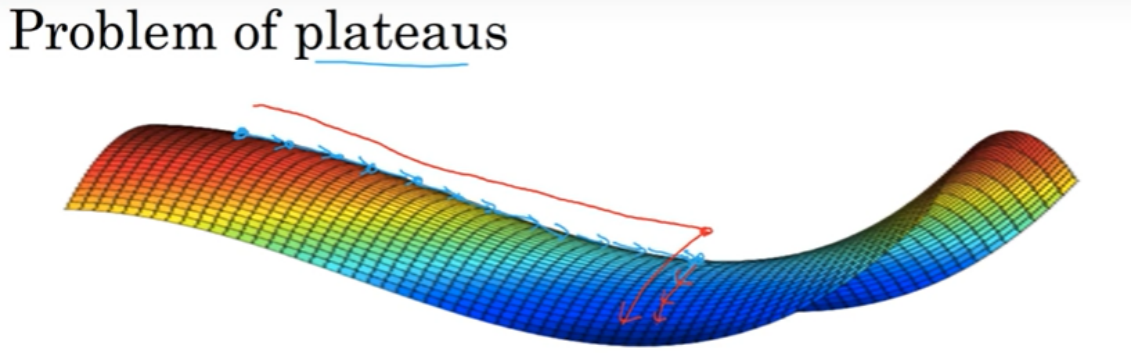


- $\alpha = 0.95^{epoch\ num}\alpha_0$ (exponential decay 라고 부릅니다.)
- $\alpha = \frac{k}{\sqrt{epoch\ num}}\alpha_0$
- $\alpha = \frac{k}{\sqrt{batch\ num}}\alpha_0$
- step 별로 α 다르게 설정

지역 최적값 문제



고차원 비용함수에서 경사가 0인 경우는 대부분 지역 최적값이 아니라 대개 안장점임.



- 안장점으로 향하는 구간인 안정지대는 미분값이 아주 오랫동안 0에 가깝게 유지되는 지역 .
- 대개 충분히 큰 Network 학습시 지역 최적값에 갇이는 일은 거의 없다.
- 안정지대의 문제점은 경사가 거의 0에 가깝기 때문에 학습속도가 느려짐. 또한, 다른 쪽으로 방향변환이 없다면 안정지대에서 벗어나기 어려움. 이는 Adam, RMSprop 등 알고리즘이 해결해줌.