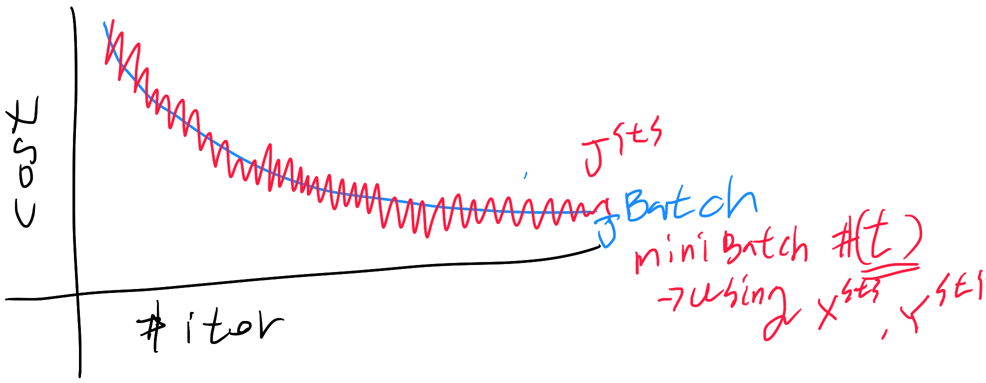
batch → SGD 에 전체 데이터 삽입
mini-batch → SGD 에 일부 데이터만

- Y 를 가지고 학습한 것과 동일할 것.

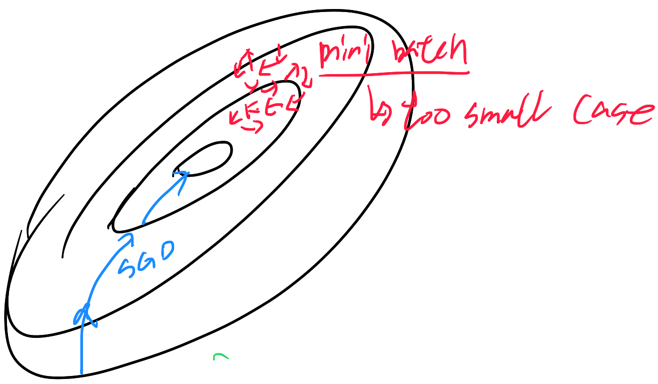
How it work?



Choosing mini-batch size.

- Size down 은 local minimum 으로 갈 확률을 높힘 Loss variation
- Huge Size 는 too Slow 의 소지가 있음 too Long per iter.

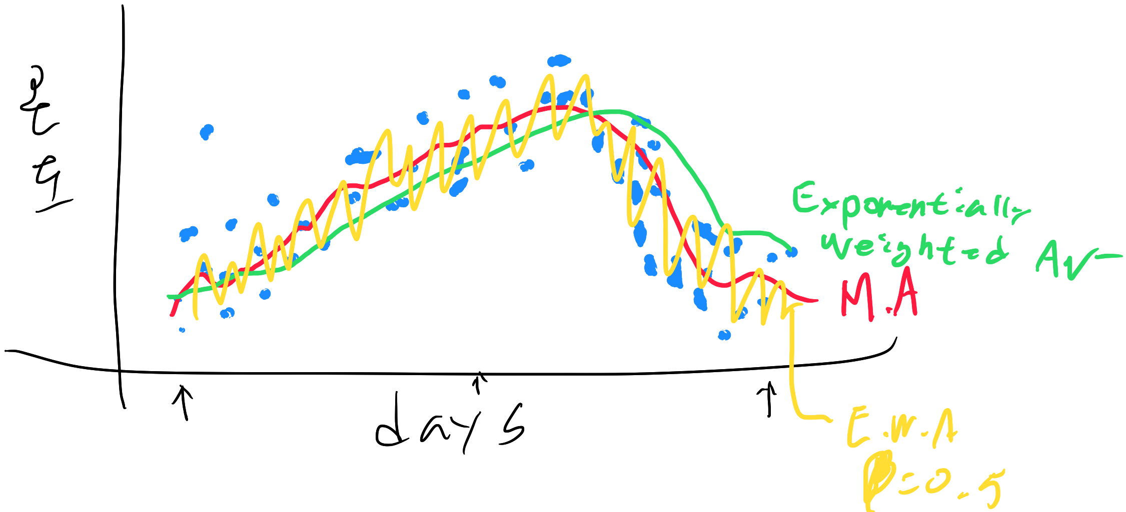
- Not too big or too small



If small traing set : use batch
(m <2000)

Typical mini-batch Size:

64, 128, 256, 512
Cpu. Gpu 메모리 허용 범위에서 크게.



$$V_0 = 0$$
$$V_1 = 0.9V_0 + 0.1\theta_1$$
$$V_2 = 0.9V_1 + 0.1\theta_2$$
$$\vdots$$

moving Average

hyper parameter.

$$V_t = \beta V_{t-1} + (1-\beta)\theta_t$$

$\beta = 0.9$: ≈ 10 days tm.

$\beta = 0.98$: ≈ 50 days tm.

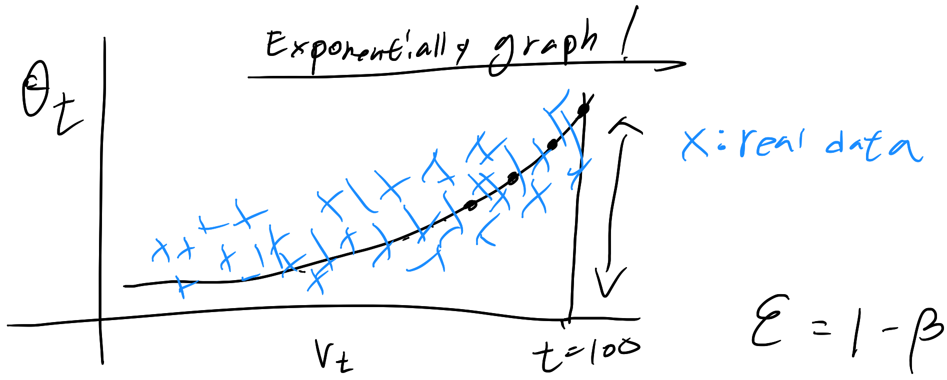
$\beta = 0.5$: ≈ 2 days tm.

more slowly
↓
Ex-
We-
Av-

Exponentially weighted averages

$$v_t = \beta_{v_t-1} + (1-\beta)\theta_t$$
$$v_{100} = 0.9v_{99} + 0.1\theta_{100}$$
$$v_{99} = 0.9v_{98} + 0.1\theta_{99}$$
$$V_{100} = 0.9v_{99} + 0.1\theta_{100}$$
$$= 0.1\theta_{100} + 0.1 \times 0.9 \cdot \theta_{99} + 0.1 \cdot (0.9)^2\theta_{98} + 0.1 \cdot (0.9)^2$$

So!



구현 이슈

V = 0

- V = Beta × V + (1-Beta) x Theta

메모리 절약 가능함.

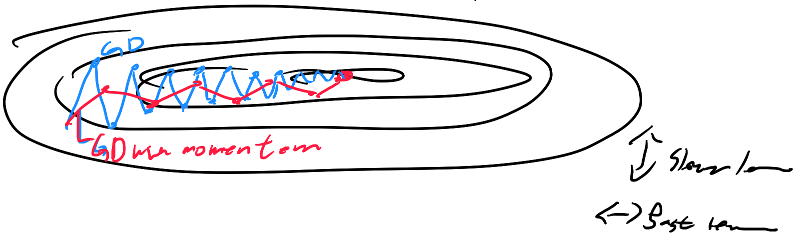
Bias correction

$$\frac{V_t}{1-\beta^t}$$

$$t = 2; 1 - \beta^t = 1 - (0 - 98)^2 = 0.396$$

Gradient descent with momentem

example of Gr— —de—



momentum:

on iteration t:

Compute dW, db on mini-batch

$$V_{dW} = \beta V_{dW} + (1-\beta) \boxed{dW} \leftarrow \text{acceleration}$$

$$V_{db} = \beta V_{db} + (1-\beta) \boxed{db}$$

Implementation details

$V_{dW} = 0$, $V_{db} = 0$

On iteration t:

Compute dW, db on the current mini-batch

$v_{dW} = \beta v_{dW} + \cancel{(1-\beta)} dW$ | $v_{dW} = \beta v_{dW} + \quad dW \leftarrow$

$v_{db} = \beta v_{db} + \cancel{(1-\beta)} db$

$W = W - \alpha v_{dW}, \quad b = b - \alpha v_{db}$

$\cancel{\frac{V_{dW}}{1-\beta^t}}$

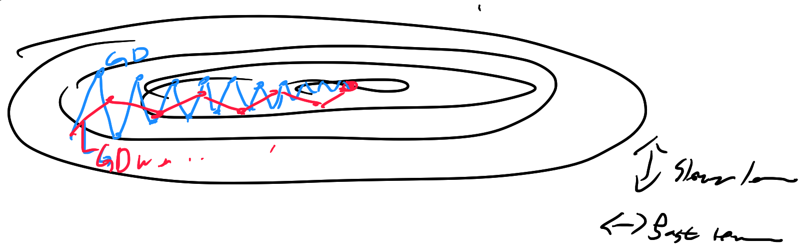
Hyperparameters: α, β $\qquad \beta = 0.9$

average over last ≈10 gradients

Andrew Ng

RMSprop

example of Gr— de—



on iteration t:

Compute dw, db on mini-batch ← element-wise

$S_{dw} = \beta S_{dw} + (1-\beta) dw^2$ ← small

$S_{db} = \beta S_{db} + (1-\beta) db^2$ ← large

$W := W - \alpha \frac{dw}{\sqrt{S_{dw}}}$

$b := b - \alpha \frac{db}{\sqrt{S_{db}}}$

Adam optimization Algorithm

$$Vdw = 0, Sdw = 0$$

Vdw is moment em

Sdw is Rms prop

$$V_{dW}^{correcte1} = V_{dW} / \left(1 - \beta^t\right)$$

$$S_{dW}^{correcte1} = S_{dW} / \left(l - \beta_2^t\right)$$

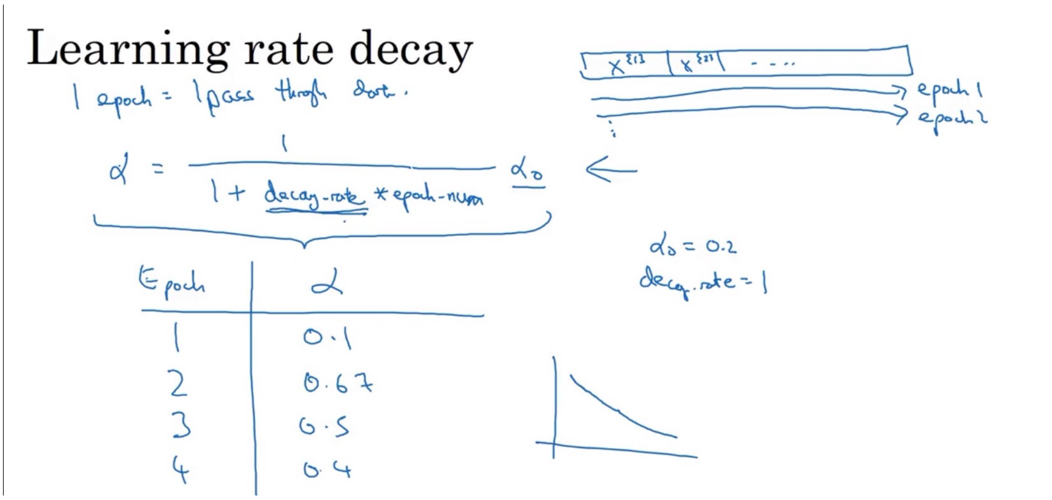
Hyperparameters

$$\beta_1 : 0.9\,(dw)$$

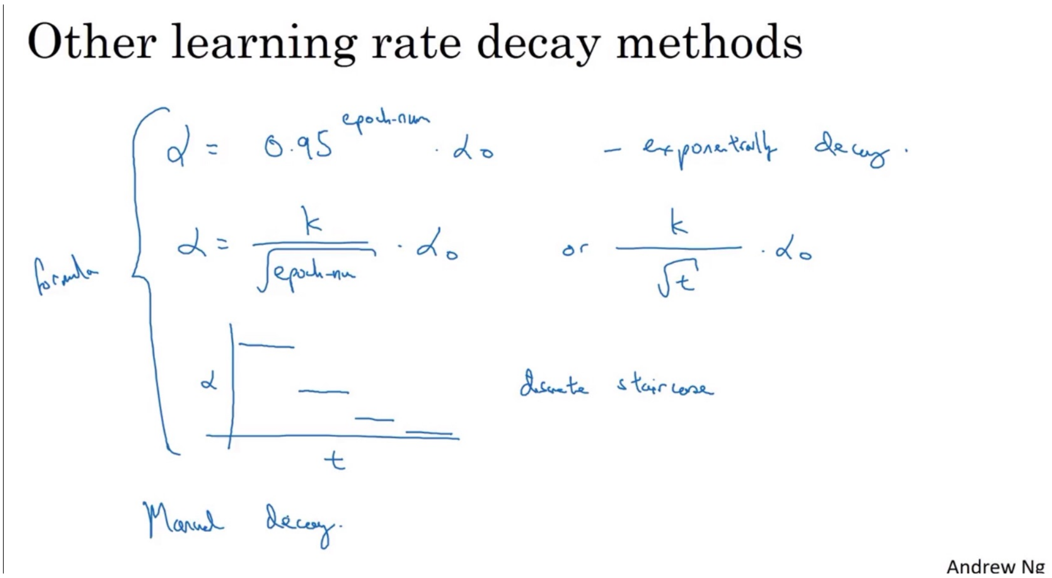
$$\beta_2 : 0 - 999\,\left(dW^2\right)$$

Learning rate decay

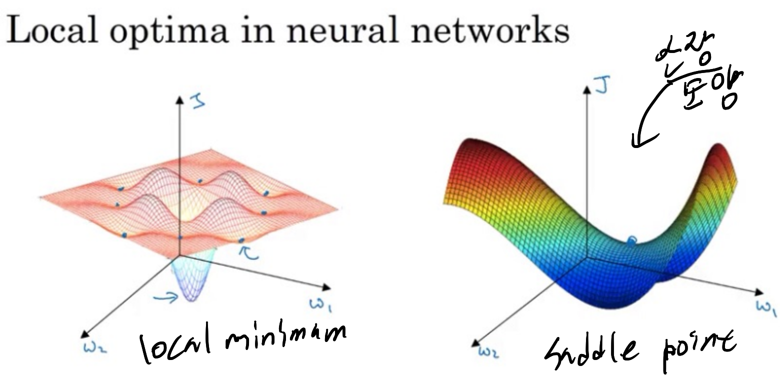
1 epoch = 1 pass through data



rate 가 너무 크면, 최적해를 못찾고
너무 작으면, 너무 오래 걸림.



Local optima

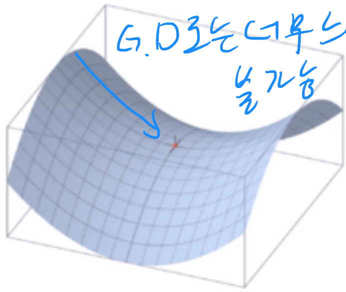


안장점

안장점(鞍裝點; saddle point)은 다변수 실함수의 변역에서, 어느 방향에서 보면 극대값이지만 다른 방향에서 보면 극소값이 되는 점이다.

수학에서 안장점이란 정류점이지만 극값을 가지지 않는 점을 말한다. 이차원의 시각에서 어느 방향에서 보면 아래로 굽어있지만 다른 방향에서 보면 위로 굽어있는 전형적인 모양이 고개 모양 혹은 안장과 닮았다고 하여 붙여졌다. 등고선의 관점에서 이차원에서의 안장점은 두 등고선의 교차점으로 나타난다.

G.D로는 너무 느리거나
불가능



함수 $f(x,y)=x^2-y^2$ 의 안장점.

목차

정의

점 $(a_1,\cdots,a_n)$이 다변수 실함수 $f(x_1,\cdots,x_n)$의 안장점이라는 것은, 영벡터가 아닌 2개의 벡터 $(M_1,\cdots,M_n)$와 $(m_1,\cdots,m_n)$가 존재하고 그 두 벡터에 대하여,

함수 $g(t)=f(a_1+tM_1,\cdots,a_n+tM_n)$가 $t=0$ 에서 극대가 되고
함수 $h(t)=f(a_1+tm_1,\cdots,a_n+tm_n)$가 $t=0$ 에서 극소가 된다

기울기 = 0

는 것이 성립한다는 것이다.

특징

미분가능한 다변수실함수의 정류점(기울기 벡터가 영벡터가 되는 점. 즉, 접평면이 수평이 되는 점)은, 안장점이 극값이다.