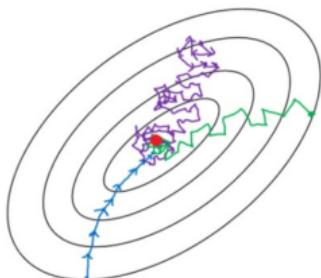


Choosing your mini-batch size

mini-batch size = m Batch gradient descent $(X^{(1)}, Y^{(1)}) \rightarrow (X, Y)$

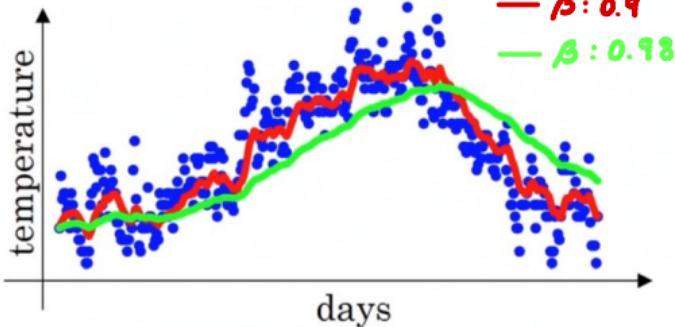
mini-batch size = 1 Stochastic gradient descent $(X^{(1)}, Y^{(1)}) = (X^{(0)}, Y^{(0)})$

$1 \sim m$ 사이의 적절한 mini-batch 사이즈 찾기



Stochastic (mini-batch size = 1)	Mini-batch	Batch (mini-batch size = m)
한번에 1개의 training set를 학습하기 때문에 빅데이터로 인해 빠른 동작을 기대할 수 없음	Vectorization + 전체 data를 학습할 때 여러번의 Step 수행	Step 당 오랜시간이 걸림

Exponentially weighted averages



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

↗ t 번째 날의 기온
 ↗ hyper parameter
 보통 0.9로 사용

$\rightarrow V_t$ 는 $\frac{1}{1-\beta}$ 기간 동안 기온의 평균을 의미

$\beta = 10$: 10 일의 기온 평균

$\beta = 0.5$: 2 일의 기온 평균

Exponentially weighted averages

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

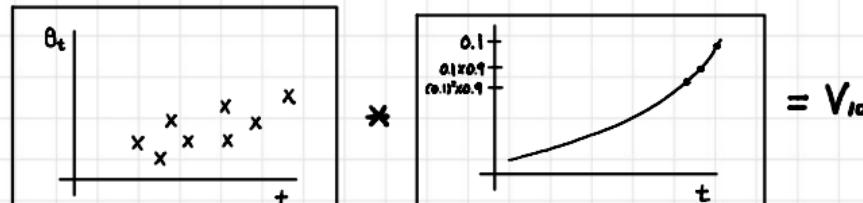
$$V_{100} = 0.9 V_{99} + 0.1 \theta_{100}$$

$$V_{99} = 0.9 V_{98} + 0.1 \theta_{99}$$

$$V_{98} = 0.9 V_{97} + 0.1 \theta_{98}$$

...

$$\rightarrow V_{100} = 0.1 \theta_{100} + 0.9 \cancel{(0.1 \theta_{99} + 0.9 \cancel{(0.1 \theta_{98} + 0.9 V_{97}})} \dots \\ = 0.1 \theta_{100} + 0.1 \times 0.9 \theta_{99} + 0.1 \times (0.9)^2 \theta_{98} + \dots$$



$$\beta = 1 - \epsilon$$

$$0.9^\infty \approx 0.35 \approx \frac{1}{e}$$

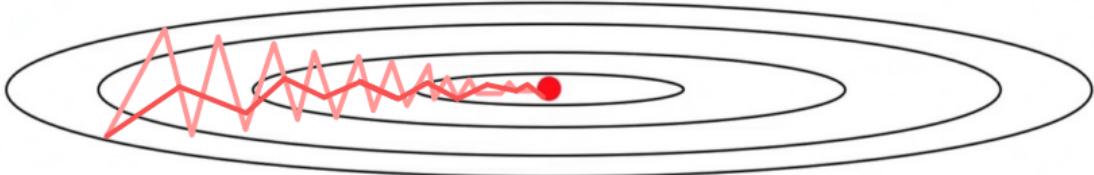
$$(1-\epsilon)^\infty = \frac{1}{e}$$

$(1-\epsilon)^\infty = \frac{1}{e}$ 를 만족시키는 η 이 기간에 되는데, 보통 $\frac{1}{e}$ 으로 구할 수 있음

→ 지수 가중 이동 평균은 구현시 아주 적은 메모리를 사용함

$\frac{V_t}{1-\beta^t}$ → 편향 보정을 사용하여 평균을 더 정확하게 계산하는 방식도 있음
(초기값과 실제 값을 비슷해지게 함)

Gradient descent with momentum



On iteration t :

Compute dW, db on the current mini-batch

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

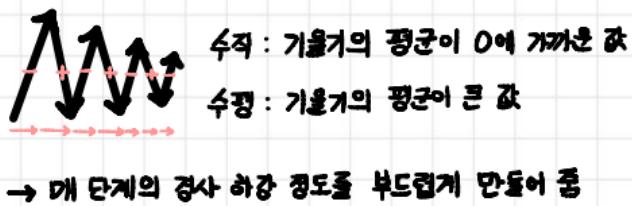
$$v_{db} = \boxed{\beta v_{db}} + (1 - \beta) \boxed{db}$$

gradient descent를
가속화시키는 역할

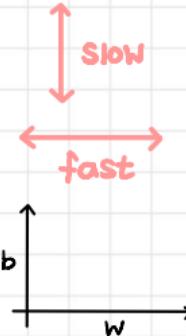
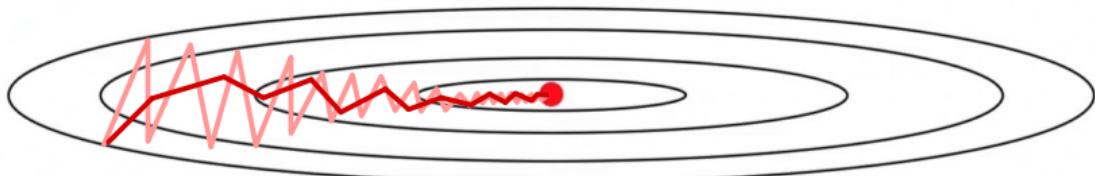
$$W = W - \alpha v_{dW}, \quad b = b - \alpha v_{db}$$

β 는 1보다 작은 수여기 때문에,
속도를 제한하는 역할 (마찰과 같은 역할)

Hyperparameters: α, β $\beta = 0.9$



RMSProp



On iteration t :

Compute dW, db on the current mini-batch

$$S_{dw} = \beta S_{dw} + (1-\beta) dW^2 \quad \text{Small}$$

element-wise

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

$$W := W - \alpha \frac{dW}{\sqrt{S_{dw}} + \epsilon} \quad b := b - \alpha \frac{db}{\sqrt{S_{db}} + \epsilon}$$

미분값이 큰 곳에서는 업데이트 시 큰 값으로 나누어주기 때문에 기존 학습률보다 작은 값으로 업데이트 → 진동을 줄여줌

미분값이 작은 곳에서는 업데이트 시 작은 값으로 나누어주기 때문에 기존 학습률 보다 큰 값으로 업데이트 → 빠른 수렴