4. 최적화 알고리즘

미니 배치 경사하강법

- **벡터화** : m개의 샘플에 대한 계산을 효율적으로 만들어 준다

명시적인 반복문 없이 훈련 세트 진행할 수 있도록 한다

but m이 매우 크다면 느리다

⇒ 훈련세트를 더 작은 훈련세트로 나눈다 : **미니배치**

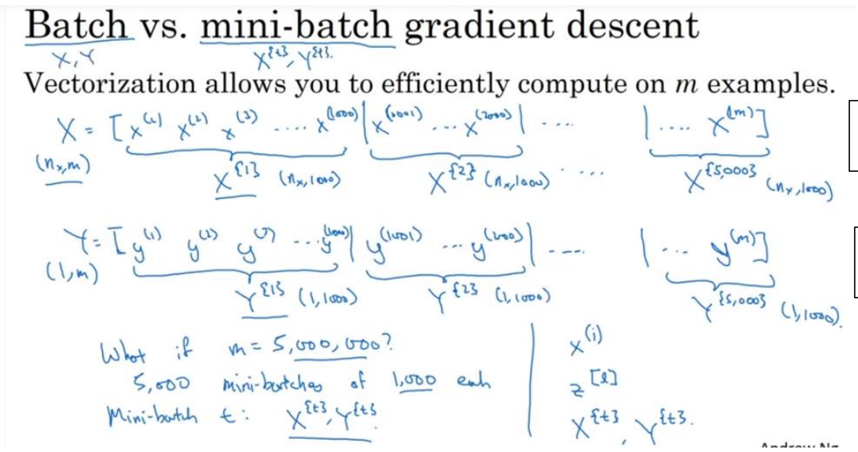
- 위첨차 표기 :

1. [l] : 서로 다른 층

2. (i) : I 번째 observation

3. {t} : 서로 다른 미니배치

<배치경사하강법 vs 미니배치경사하강법>



X{i} : (n_x,1000) 차원

Y{i} : (1,1000) 차원

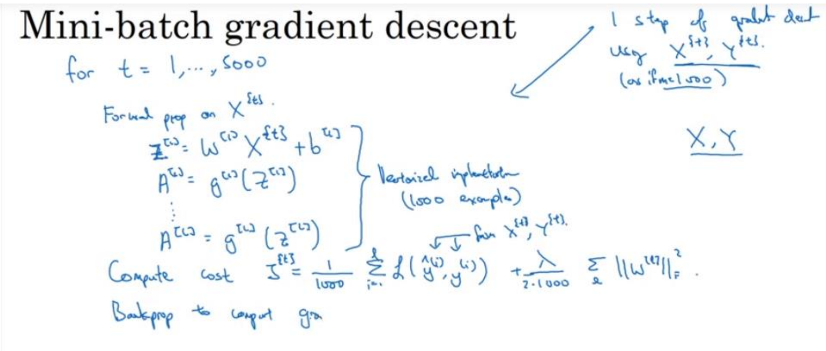
- 배치경사하강법 : 모든 훈련 세트의 모든 배치를 동시에 진행한다
- 미니배치경사하강법 : 전체 훈련 세트 X,Y 를 한번에 진행시키지 않고 X{t},Y{t}끼리 진행한다

- 1000개씩 나눈 5000개의 미니배치

<미니배치 경사하강법>

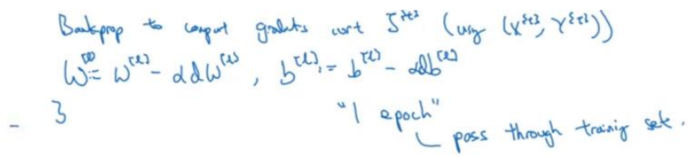
- t=1,2, … , 5000
- X{t},Y{t}를 이용하여 1step의 경사하강법 진행한다
- 백터화를 사용해 모든 샘플을 한번에 진행한다

정방향 전파



- Z[1]=W[1]X{t}+b[1]
- A[1]=g[1](Z[1]) .. A[l]=g[l]Z[l]
- 비용함수 계산 : J{t}=1/1000 sum(L(yhat(i),y(i)) + 정규화항
-

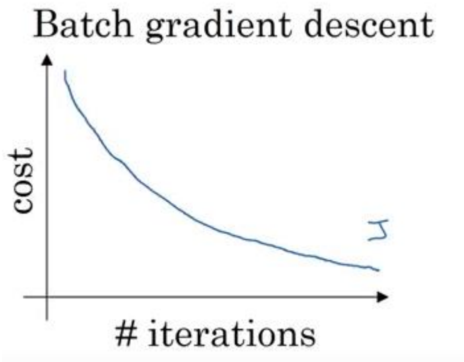
역전파

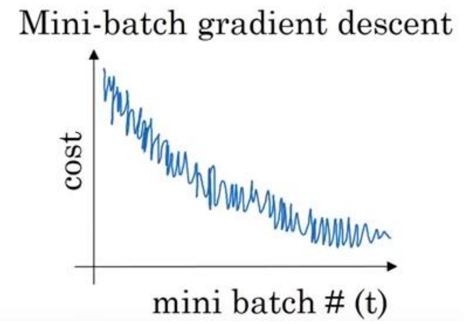


- 업데이트 : W[l]=W[l]-alpha*dW[l] ; b[l]=b[l]-alpha*db[l]
- 미니경사하강법을 사용한 훈련세트를 지나는 한번 반복

  = 1 **에포크** (슨련세트를 거치는 반복의 수 의미)
- 미니배치경사하강법 한 반복은 5000개의 경사하강 단계 거친다

미니배치 경사하강법 이해하기





배치 경사 하강법에서는 한 번의 반복을 돌 때마다 비용 함수의 값은 계속 작아져야 한다

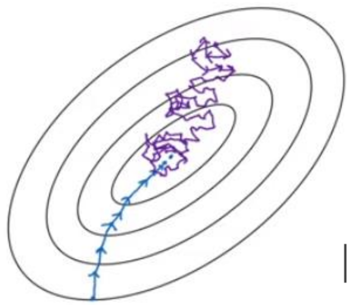
미니배치 경사 하강법에서는 전체적으로 봤을 때는 비용 함수가 감소하는 경향을 보이지만 많은 노이즈가 발생한다 - J{t} by X{t},Y{t}

- 노이즈가 발생하는 이유 :

  X{1} Y{1}은 쉬운 미니배치라서 비용이 낮은데, X{2},Y{2}가 더 어려운 미니배치일 경우

<미니배치 사이즈 결정하기>

- 사이즈 = m일 때 : 일반적인 경사하강법
- 사이즈 = 1일 때 : 확률적 경사 하강법 (각각의 샘플은 하나의 미니배치)



경사하강법 : 노이즈가 적고 상대적을 큰 단계step를 취한다

확률적 경사하강법 : 모든 반복에서 하나이 훈련 샘플 이용하니까 잘못된 방향을 가르켜 잘못된 곳으로 갈 수 있고 극단적으로 노이즈가 많아서 절대 수렴하지 않는다

⇒ 실제로 사용하는 사이즈 : 1과 m 사이

배치경사하강법 (미니배치 사이즈 m) => 큰 훈련세트를 모든 반복에서 사용하여 시간이 오래 걸린다

확률적 경사하강법 => 간단하고 노이즈도 작은 학습률을 이용해서 줄일 수 있다 근데 벡터화에서 얻을 수 있는 속도 향상을 잃는다

<in-between 의 장점>

1. 많은 벡터화를 얻어서 속도가 빨라진다
2. 전체 훈련세트가 진행되기를 기다리지 않고 진행할 수 있다
3. 더 일관되게 전역의 최소값으로 향하고, 작은 영역에서 수렴하거나 진동한다
4. 2의 제곱수로 실행해보고 값 선택하기

<미니배치 사이즈를 결정하는 가이드라인>

1. 작은 훈련세트이면 그냥 배치 경사하강법 사용하기 (2000보다 작은 세트)
2. 큰 훈련 세트이면 미니배치 크기는 64~512 가 일반적이다
3. 미니배치에서 모든 X{t},Y{t} 가 CPU 메모리에 맞는지 확인해야한다

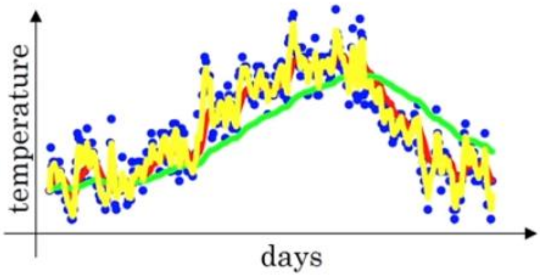
지수 가중 이동 평균

v0 = 0 으로 초기화

v1 = 0.9v0+0.1세타1 ; v2 = 0.9v1+0.1세타2 ; ...

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

⇒ vt ~~ 1/(1-β) * 일별 기온의 평균



-

- β=0.98 (이전 값에 많은 가중치를 준다는 뜻) => 더 매끄러운 곡선 but 곡선이 올바른 값에서 멀어진다
- β=0.95 =>노란색. 2일의 기온을 평균해서 노이즈 많고 이상치에 민감. 빠르게 적응

★ 하이퍼 파라미터 값을 바꿈으로써 다른 효과를 얻는다

지수가중이동평균 이해하기

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

연속적으로 대입하면 :

⇒ $v_{100}=0.1\theta_{100}+0.1\times 0.9\theta_{99}+0.1\times(0.9)^2\theta_{98}+\cdots$ = 가중치의 합!

v100 : 두 함수간에 요소별 곱셈을 해서 더하면 구할 수 있다

- 앞에 곱해지는 계수들을 더하면 (편향 보정) 1에 가까워진다

- 얼마의 기간이 이동하면서 평균이 구해졌는가?

β=(1−ε) 라고 정의 하면

(1−ε)^n=1/e를 만족하는 n : 기간 = 보통 1/ ε

처음에는 가중치가 1/e보다 커지다가 감소가 가파르게 일어난다

ex) 기간 50이면 1/(1- β) ε =1- β =>_평균적인 온도가 몇일이 될 지 알려준다

$$V_\theta := 0$$
$$V_\theta := \beta v + (1-\beta)\theta_1$$
$$V_\theta := \beta v + (1-\beta)\theta_2$$
$$\vdots$$

$\rightarrow V_\theta = 0$

Repeat {

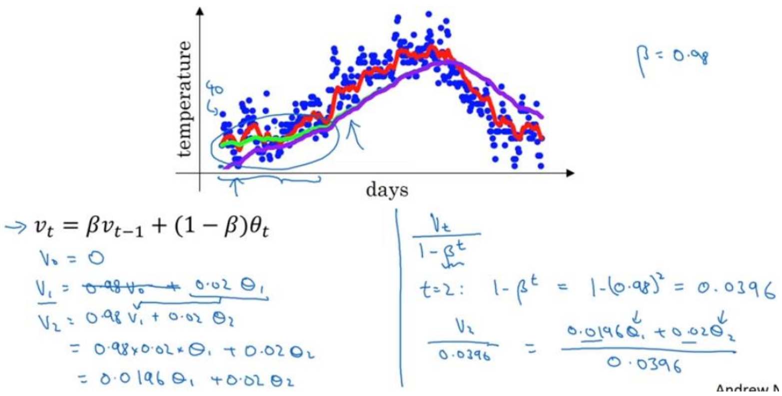
Get next θ_t

$V_\theta := \beta V_\theta + (1-\beta)\theta_t \leftarrow$

}

- 지수 가중 이동 평균의 장점은 구현시 아주 적은 메모리를 사용한다는 것이다

지수 가중 이동 평균의 편향 보정

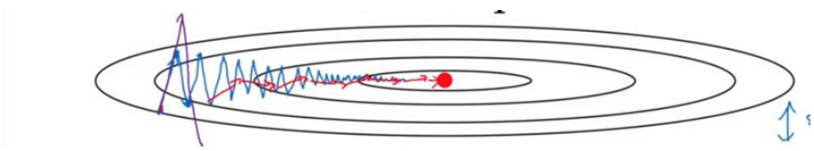


편향 보정

세타1과 세타2의 가중평균에 편향을 없앤 값이 된다

- 편향 보정으로 평균을 더 정확하게 계산할 수 있다.
- 보라색곡선이 매우 작은 값에서 시작하기 때문에 편향 보정을 이용한다
- 추정 초기단계에서 보정한다
- **v_t/(1- β _t)**
- t가 커지면 β t는 0에 가까워진다 => 편향 보정은 효과가 거의 없어진다

momentum 최적화 알고리즘



- 일반적인 경사하강법보다 <u>빠르게 작동한다</u>
- <u>경사에 대한 지수가중평균을 계산하고 그 값으로 가중치를 업데이트 한다</u>
- 학습률이 너무 크지 않아야 진동이 커지는 것을 막을 수 있다
- 수직축에서는 진동을 막기 위해 학습이 느리게 되길 원하고 수평축에서는 빠른 학습을 원한다
- 반복 t에서 현재의 미니배치에 대한 보편적인 도함수 dw, db로 계산.

VdW=βVdW+(1−β)dW ; Vdb=βvdb+(1−β)db

w:=w−αVdW ; b:=b−αVdb

- 경사하강법의 단계를 부드럽게 만들어준다
- 수직 방향의 진동이 0에 가깝게 평균이 만들어진다
- 수평 방향에서 모든 도함수는 꽤 큰 값을 가진다

- 그릇 모양을 최소화하려면 도함수의 항들은 가속을 제공하고 모멘텀항은 속도를 나타내고 β는 1보다 작기때문에 마찰을 제공한다
- β 일반적 = 0.9
- 편향 보정 : v_dw / (1- β^t) : 잘 사용하지 않는다 (편향 추정이 잘 일어나서)
- *v_dw= β*v_dw+dw 라고 하면 => 1/(1- β) 계수로 스케일링 된다 : 알파가 대응하는 값으로 바뀔 필요 없다*

RMSProp 최적화 알고리즘

- root mean square prop : 경사하강법 빠르게 하는 다른 방법
- 수직축 b, 수평축 w라고 하면 b방향 학습속도 늦추고, w방향 학습 속도 빠르게 하고 싶다
- t반복에서 현재의 미니배치에 대한 도함수 dw, db 계산
- 지수가중 평균을 유지하기위한 표기 **: s_dw = β*s_dw+(1- β)*dw^2 (요소별 제곱 의미)**
- 도함수의 제곱을 지수가중평균하는 것이다
- **RMSprop 업데이트** : $w := w - \alpha \frac{dW}{\sqrt{S_{dW} + \epsilon}}$
- s_dw 가 작고, db가 큰 숫자이기를 원한다 (실제로 수직 방향의 도함수(db)가 수평 방향의 도함수(dw)보다 크다) => dw^2는 상대적으로 더 작다
- **수직 방향에서는** 더 큰 숫자로 나눠서 업데이트 (업데이트 감소) 하기 때문에 진동을 줄인다
- **수평방향에서는** 작은 숫자로 나눠서 업데이트한다 (계속 나아간다)
- 효과 : 큰 학습률을 사용해 빠르게 학습하고 수직 방향으로 발산하지 않는다
- 실제로는 차원의 벡터로 생각한다
- 0으로 나뉘지 않게 주의해야한다

Adam 최적화 알고리즘

- Adam 최적화 알고리즘 = RMSprop+모멘텀
- VdW=0,SdW=0 로 초기화한다
  - 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: $w := w - \alpha \frac{V_{dW}^{correct}}{\sqrt{S_{dW}^{correct}} + \epsilon}$

모멘텀 업데이트

RMSprop 업데이트

RMS 부분 추가

<하이퍼파라미터>

- alpha : 다양한 값 시도해서 찾기
- β _1 : 0.9 가중이동평균
- β _2 : 0.999 dw^2 와 db^2 의 이동가중평균 계산한 것
- ε : 10^(-8) 근데 크게 상관 없다

adaptive moment estimation

- β _1 : 도함수의 평균 계산 (첫번째 모멘트)
- β _2 : 지수가중평균의 제곱을 계산 (두번째 모멘트)

학습률 감쇠

- 시간에 따라 학습률을 천천히 줄여서 학습 알고리즘의 속도를 높일 수 있다
- 작은 미니배치에 대해 미니배치경사하강법을 할 때 노이즈 때문에 정확하게 최소값에 도달하지 못한다
- alpha 가 작아지면 => 단계마다 진행정도가 작아져서 최솟값 주변의 밀집된 지역에서 위치하게 된다
- 초기에는 큰 스텝, 이후에는 작은 스텝으로 진행하는 것이 좋다

- **alpha = 1/(1+decay_rate)*epoh_num**
- 에포크 수에 대한 함수에서 학습률은 점차 감소한다

지수적 감쇠

- alpha<1
- alpha = 0.95^epoch_num*alpha_0

이산 계단

직접 조작하는 감쇠 (작은 모델일 때)

- $\alpha = \frac{1}{1 + decay\ rate\ \times epoch\ num}\alpha_0$
- $\alpha = 0.95^{epoch\ num}\alpha_0$ (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